

THESIS / THÈSE

MASTER EN SCIENCES INFORMATIQUES

Implantation d'une gestion de barèmes

Evrard, Etienne

Award date:
1986

Awarding institution:
Universite de Namur

[Link to publication](#)

General rights

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal ?

Take down policy

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

**Implantation d'une gestion
de barèmes**

E. Evrard
F.U.N.D.P.
3e Licence et maîtrise
en Informatique
Année académique 1985-1986

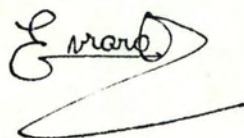
Promoteur R. Lesuisse
F.U.N.D.P.

Responsable extérieur P. Taton
Crédit Communal de Belgique

Je remercie vivement ,
Monsieur Lersuisse , Promoteur de mon mémoire ,
pour son aide précieuse et ses directives
assidues dans la réalisation de ce travail .

Je présente également mes
remerciements chaleureux à Monsieur Pascal Taton
pour sa collaboration active et dévouée, ainsi
qu'au personnel du Crédit Communal de Bruxelles .

E. EVRARD

A handwritten signature in dark ink, appearing to read 'Evrard', with a long, sweeping horizontal stroke extending to the right.

P L A N

CHAPITRE I : Introduction.

1.1 Le problème.

1.2 Définition et exemples de barèmes.

CHAPITRE II : Analyse fonctionnelle.

2.1 Introduction.

a) Premier exemple.

b) Deuxième exemple.

c) Les traitements.

2.2 Définition des concepts "clé" du problème.

2.3 Schéma Entités - Associations et ses contraintes.

a) Rappel théorique.

b) Le schéma.

c) Dictionnaire des types d'entités, des types d'associations et de leurs attributs.

d) Les contraintes d'intégrité du schéma.

2.4 Description des traitements.

a) Les traitements interactifs.

b) Les traitements par programmes.

CHAPITRE III : Analyse d'implémentation.

3.1 Schéma de la base de données.

a) Formation du schéma d'accès à partir du schéma E.A.

b) Le schéma.

3.2 Architecture du logiciel.

- a) Le schéma modulaire et les règles qui président à sa constitution.
- b) Description des modules.
 - Le module BAREME.
 - Le module ES (Entrées-sorties).
 - Le module SECURITE.
 - Le module DOCUMENTATION.
 - Le module gestion interactive de barèmes.
 - Le module gestion de barèmes par programmes.

CHAPITRE IV : Les performances.

CHAPITRE V : Conclusions.

Bibliographie.

LES ANNEXES.

Annexe 1: Les algorithmes principaux.

Annexe 2: Les programmes.

Annexe 3: Les écrans.

CHAPITRE I

Introduction

1.1 Le problème.

Dans la réalisation de programmes financiers, plusieurs responsables et programmeurs ont remarqué la nécessité d'accéder constamment à ce qu'ils appellent des "barèmes". On appellera "BAREME", une table de valeurs ou de renseignements distincts de même type créée pour les besoins de l'entreprise. Chaque accès inclut l'obligation pour le programme qui le réalise, de connaître toute la structure de ces barèmes.

De là, jaillit l'idée d'avoir une gestion de barèmes tout à fait indépendante des programmes existants.

Certaines personnes se penchèrent sur le problème et se rendirent très vite compte de la difficulté que celui-ci présentait. D'où l'idée de le proposer comme mémoire d'étudiant.

L'objectif de notre mémoire est donc d'implémenter un système de gestion de barèmes que des programmes d'application puissent utiliser sans connaître la manière dont il gère les barèmes. Lorsqu'un tel programme interrogera notre système, il ne pourra attendre longtemps sa réponse, il faudra donc viser la performance. Il est également souhaité d'avoir la possibilité d'effectuer certains travaux de consultation et de création en mode interactif, et de tenir compte d'une certaine sécurité. Qui a accès au système ? A quels barèmes un utilisateur a-t'il accès ? Qu'a-t'il modifié ? A-t'il le droit de modifier un barème ?

Il faudrait donc étudier et réaliser une structure de données pour la gestion de barèmes, et définir des primitives d'accès à cette structure telles qu'on puisse les utiliser en ignorant totalement sur quelle sorte de structure ces primitives travaillent.

Abordons maintenant de manière plus précise ce qu'est un barème.

1.2 Définition et exemples de barèmes.

Un barème est donc un répertoire de valeurs distinctes d'un même type créé pour les besoins de l'entreprise. Sur ce répertoire on définit un code afin de représenter ces valeurs de manière plus compacte.

Un barème peut être simple ou composé.

Un barème simple se résume donc en une colonne de valeurs de code associées à leur(s) signification(s) c-à-d leur(s) libellé(s).

Un barème composé est un tableau de plusieurs colonnes, chacune contenant des valeurs d'un certain type. Une ligne de ce tableau possède une certaine signification et est identifiée par une valeur de code.

Voyons plus précisément, dans quelques exemples ce dont il s'agit.

a) Barèmes simples :

- Barème des régimes linguistiques

<u>Code</u>	<u>Signification</u>
FR	français
AN	anglais
NR	néerlandais
AL	allemand

- Barème des codes taux

<u>Code</u>	<u>Signification</u>
0	code taux normal
1	code taux sans intérêt
2	code de faveur
3	code taux spécial gros montants

- Barème des types de comptes

<u>Code</u>	<u>Signification</u>
CV	compte à vue
CD	carnet de dépôt
CP	compte de placement
CT	carnet à terme

- Barème des types d'intérêts

<u>Code</u>	<u>Signification</u>
CR	créditeur
DB	débiteur

- Barème des taux d'intérêts

<u>Code</u>	<u>Signification</u>
0,00%	pas d'intérêt
0,50%	un demi pour-cent
2,00%	deux pour-cent
6,00%	six pour-cent
9,00%	neuf pour-cent

b) Barème composé :

Barème des codes taux en compte

<u>Code</u>	<u>Code taux</u>	<u>Type de comptes</u>
1	0	CV
2	3	CT
3	2	CD
4	2	CP

Signification de la ligne de code 1 : sur un compte à vue, on applique un code taux normal.

Signification de la ligne de code 2 : sur un carnet à terme, on applique un code taux spécial gros montants.

Signification de la ligne de code 3 : sur un carnet de dépôt, on applique un code taux de faveur.

Signification de la ligne de code 4 : sur un compte de placement, on applique un code taux de faveur.

Un barème peut donc être vu soit comme une table de libellés, soit comme un petit fichier.

C H A P I T R E I I

Analyse fonctionnelle

2.1 Introduction.

Le but de ce chapitre est d'introduire le lecteur sur base de deux exemples, à la structure de données, aux concepts et aux traitements relatifs au problème posé.

a) Premier exemple.

Soit le barème des types de comptes.

Identificateur: IS0001

Code	Signification(s)
CV	compte à vue, ziektrekening
CD	carnet de dépôt, depositoboekje
CP	compte de placement
CT	carnet à terme

(Fig 2.1)

Ce barème, créé à une certaine date, est bien une table de valeurs distinctes d'un même type. Comme on peut le voir, ce barème concerne les différents types de comptes d'une banque. Dans un tel organisme, il est fort probable qu'on ne parle pas uniquement le français. Pour cette raison, on aurait très bien pu écrire le nom de ce barème en néerlandais, en anglais, ou encore en allemand. Ce sont les quatre langues dont on peut avoir besoin en Belgique.

Penchons nous maintenant sur le contenu de ce barème. On peut y trouver deux colonnes. L'une comportant un certain nombre de libellés, l'autre en regard contenant des valeurs de code associées à ces libellés. On peut constater qu'à une même valeur de code peut correspondre un ou plusieurs libellés. Voyons que pour la valeur de code

CV, on peut y lire deux libellés, un en français, l'autre en néerlandais. Plusieurs langues peuvent en effet être utiles vu que des personnes de régimes linguistiques différents peuvent travailler dans cette même entreprise, ici le crédit communal de Belgique.

Proposons-nous maintenant d'observer ce barème dans le temps!

Actuellement, dans la banque, il existe ce qu'on appelle les "carnets à terme". Mais demain, ceux-ci existeront-ils encore? Il est possible que les responsables décident un jour de supprimer cette sorte de carnet car elle ne présente plus assez d'intérêt ni pour eux, ni pour le client. Eventuellement, elle pourrait être remplacée par une autre plus avantageuse.

Donc, par exemple, la valeur de code CT, créée à une certaine date, et son libellé seront utilisés dans ce barème pendant une certaine période. Celle-ci se matérialise par une date et une heure de début d'utilisation, ainsi qu'une date de fin d'utilisation de cette valeur. Au delà de cette période, ils seront supprimés du barème ou encore conservés par sécurité pendant une nouvelle période exprimée en nombre de jours. De toute manière, après celle-ci, la suppression doit avoir lieu. Précisons qu'une valeur de code remplacée par une autre est appelée une "version successive", et que ce nombre de versions successives d'une valeur est limité afin de ne pas retenir pour rien des valeurs trop anciennes.

Au C.C.B., certaines personnes s'occupent de définir des codes associés aux barèmes. Ils exigent que toutes les valeurs d'un code soient d'un même type (alphabétique, alphanumérique, ou numérique) et ne dépassent pas une certaine longueur. Un code a pour but de représenter les informations de manière plus compacte. On les utilise couramment dans les banques.

Il est aussi possible qu'un même barème tel que celui montré à la Fig 2.1 puisse faire l'objet d'une ou plusieurs documentations

par son créateur. Ce dernier doit éventuellement pouvoir dire pourquoi il l'a créé: dans quel cadre, quelle en est l'utilité dans la banque, ou encore quelles en sont les remarques éventuelles, etc.

Chaque valeur de code doit également pouvoir faire l'objet de telles documentations. Ces dernières possèdent un ou plusieurs titres exprimés dans différentes langues et se composent en fait d'un texte structuré en paragraphes. Chaque paragraphe est lui-même organisé en un certain nombre de lignes (24 au maximum), possède un numéro identifiant et un numéro d'ordre dans sa documentation. Chaque ligne possède un numéro d'ordre dans son paragraphe.

Dans la banque, on voudrait que tout le personnel n'ait pas nécessairement accès à un barème. Certains sont plus confidentiels que d'autres ou uniquement destinés à des chefs. Pour résoudre ce problème, le créateur du barème désignera lui-même les personnes ou groupe de personnes qui ont les droits d'accès et/ou de modification à son barème. De plus, on voudrait toujours pouvoir retrouver la personne qui a modifié une valeur de ce barème. Une modification étant bien entendu autorisée si on en possède le droit. Pour être complet, le groupe des utilisateurs du système devra en plus posséder un chef qui seul aura le droit de créer et de supprimer des utilisateurs. Bien entendu, pour pouvoir entrer dans le système, il faudra faire partie du groupe des utilisateurs. Le système devra donc identifier l'utilisateur par un nom et un mot de passe.

Modifier une valeur, c'est en fait en créer une nouvelle et la relier à l'ancienne afin de toujours pouvoir dire à partir de la nouvelle, quelle est celle qui a été remplacée. Une valeur peut donc en remplacer une autre, qui elle-même pourrait en faire de même... Cela en n'excédant bien sûr jamais le nombre maximum de versions successives que l'on peut conserver.

Nous venons ici de parcourir de manière détaillée les articula-

tions et les contraintes qui jouent sur un tel barème. L'exemple suivant va maintenant nous permettre de découvrir les faits et les contraintes qui restent dans le problème posé.

b) Deuxième exemple.

Soit le barème des taux d'intérêts à appliquer en compte.

Identificateur: IS0005

<u>Code</u>	<u>Type de comptes</u>	<u>Code taux</u>	<u>Type d'intérêts</u>	<u>Taux d'intérêts</u>
1	CV	0	CR	2 %
2	CT	3	CR	9 %
3	CD	2	DB	2 %
4	CP	3	CR	9 %

(Fig 2.2)

Signification de la ligne de code 1 : sur un compte à vue, de code
taux normal et de type d'intérêts créditeur, on applique un taux
d'intérêts de 2 %.

Signification de la ligne de code 2 : sur un carnet à terme, de code
taux spécial gros montants, et de type d'intérêts créditeur, on ap-
plique un taux d'intérêts de 9 %.

Signification de la ligne de code 3 : sur un carnet de dépôt, de code
taux de faveur, et de type d'intérêts débiteur, on applique un taux
d'intérêts de 2 %.

Signification de la ligne de code 4: sur un compte de placement de
code taux spécial gros montants, et de type d'intérêts créditeur, on
applique un taux d'intérêts de 9 %.

Ces différentes significations tout comme le nom du barème au-
raient pu être exprimés aussi dans d'autres langues.

Tout comme l'exemple précédent, ce barème possède des valeurs de
code associées à leurs significations. Mais en fait, ces significa-
tions concernent aussi ici les quatre colonnes supplémentaires de

la Fig 2.2. Quelles sont donc ces quatre autres colonnes ? Si on prend la colonne "Type de comptes", on constate qu'elle contient en fait des valeurs de code appartenant à un autre barème. Et c'est la même chose pour les trois autres.

Donc, on peut remarquer que la valeur de code 1 de notre barème est reliée aux valeurs de code "CV" du barème des types de compte, "0" du barème des codes taux, "CR" du barème des types d'intérêts, et "2 %" du barème des taux d'intérêts. Ce barème est donc composé de quatre autres barèmes. Après maintes réflexions, on a pu constater qu'un tel barème pouvait toujours être composé de barèmes à deux colonnes telle qu'à la Fig 2.1. Ceci est donc une contrainte importante du système. De plus, pour qu'une ligne de ce barème soit utilisable, les quatre valeurs de code des quatre barèmes composants doivent être dans leurs périodes d'utilisation au moment de la création de cette ligne.

Reprenons maintenant, dans ce barème, la ligne dont la valeur de code est 1. Toutes les valeurs de code auxquelles elle est reliée possèdent, comme on l'a vu dans le premier exemple, une période d'utilisation. Afin d'empêcher que, par exemple, la valeur de code "CV" ne soit plus utilisable alors que notre ligne le soit toujours (ce qui serait inacceptable), il faut donc que la période d'utilisation de cette ligne soit limitée à la période la plus petite de celles des valeurs de code auxquelles elle est reliée.

On a également vu dans l'exemple précédent qu'une valeur de code qui n'est plus utilisable, peut encore être conservée pendant une période exprimée en nombre de jours. Une ligne de notre barème le peut aussi et cette période est limitée à la période la plus petite de celles des valeurs de code auxquelles elle est reliée.

Ces différentes contraintes sont donc valables dans le cas où l'on a un barème plus "compliqué" comme à la Fig 2.2.

c) Les traitements.

Voici maintenant les différentes opérations que l'on voudrait implanter dans notre système.

1/ Créations :

- Créer un barème (y compris les droits d'accès et de modification).
- Créer une valeur dans un barème.
- Créer une documentation concernant un barème.
- Créer une documentation concernant une valeur dans un barème.
- Créer un nouvel utilisateur dans le système (opération réservée au chef du système).

2/ Consultations :

- Consulter un barème.
- Obtenir les libellés d'une valeur de code dans un barème.
- Obtenir toutes les lignes d'un barème qui contiennent N valeurs de code (recherche selon critère).
- Obtenir le nom de tous les barèmes du système.
- Un utilisateur doit pouvoir obtenir tous les noms des barèmes auxquels il a accès.
- Obtenir la composition d'un barème.
- Obtenir les valeurs qu'un utilisateur a modifiées.
- Consulter une documentation.
- Etc. (on peut encore en imaginer beaucoup, mais il faut bien se limiter)

3/ Modifications :

- Pouvoir modifier une valeur dans un barème.

4/ Suppressions :

- Pouvoir nettoyer le système c-à-d supprimer toutes les valeurs qui ne peuvent plus servir, et tous les barèmes qui ne contiennent plus de valeur.
- Pouvoir supprimer une documentation.
- Pouvoir supprimer un utilisateur (opération réservée au chef du système).

Le système qu'on désire réaliser sera un prototype et on n'aura sans doute pas le temps de réaliser l'entièreté de ces opérations. Pour cela, un choix de réalisation sera nécessaire.

Qui questionnera le système ? Ce sera soit un programme soit un utilisateur en mode interactif. Les questions primordiales seront celles posées par programmes car ces derniers doivent obtenir leurs réponses beaucoup plus vite qu'un utilisateur interactif. L'exécution d'un programme ne peut en effet pas être longtemps interrompue. L'accès à réaliser pour répondre à ce genre de questions sera donc à optimiser. Les opérations que l'on peut lancer par programmes sont : - Obtenir le(s) libellé(s) d'une valeur de code;

- et ce qu'on a appelé la "recherche selon critère".

En ce qui concerne le mode interactif, toutes les opérations sont concernées. On essaiera dans la mesure du possible de ne pas faire attendre l'utilisateur.

Ceci constitue une première approche intuitive de ce que sera le système qu'on se propose de réaliser et donc du genre de choses que le système devra prendre en compte.

2.2 Définition des concepts "clé" du problème.

Après avoir vu plus en détail les différents éléments qui interviennent dans le problème, on peut maintenant définir, un à un, les concepts qui sont relatifs à l'application que l'on va développer.

a) Le concept de " Valeur de code"

Une valeur de code est une valeur qui a été choisie en vue de représenter une information c-à-d une signification de manière plus compacte.

b) Le concept de "libellé"

Une signification d'une valeur de code sera dorénavant appelée "libellé" de cette valeur de code. Comme on l'a déjà vu, une signification peut devoir s'exprimer par exemple dans plusieurs langues. Donc une même valeur de code pourra toujours posséder plusieurs libellés si cela est nécessaire dans l'entreprise.

Les noms que l'on donne à un barème seront également appelés "libellés" de ce barème.

c) Le concept de " Valeur de barème" (Valeur simple et composée)

On définit une valeur de barème par sa constitution. Elle comprend donc:

- Une date de création,
- Une date de début d'utilisation,
- Une heure de début d'utilisation,
- Une date de fin d'utilisation,
- Une valeur de code,
- Un ou plusieurs libellés associés à cette valeur de code

Si une valeur de barème comprend uniquement les éléments cités ci-dessus, cette valeur sera dite "simple".

Une valeur de barème sera dite "composée" si elle comprend en plus une liste de valeurs de code composantes.

d) Le concept de "barème" (simple, composé, composant)

Un barème simple est une liste de valeurs simples comme définies au point c.

Un barème composé est une liste de valeurs composées comme définies au point c.

Un barème composé peut encore être défini comme la combinaison de N barèmes simples, chaque ligne contenant, en fait, la valeur de code identifiant la ligne, et N autres valeurs de code appartenant respectivement aux N barèmes simples qui le composent. Ces derniers barèmes sont alors aussi appelés "barèmes composants". Il est à remarquer que si un barème est composé, il ne peut l'être que de barèmes simples. Cette contrainte est à respecter strictement dans le système.

Chaque barème possèdera aussi une date de création, un numéro identifiant, un type et une longueur maximum pour ses valeurs de code.

e) Le concept de valeur de barème "d'actualité"

Une valeur de barème sera dite "d'actualité" si la date du jour courant se trouve entre la date de début et celle de fin d'utilisation de cette valeur.

La période qui s'écoule entre ces deux dates sera aussi appelée "période de validité" de la valeur. Avant cette période, une valeur de barème n'est pas encore d'actualité. Et après celle-ci, cette valeur n'est plus d'actualité.

f) Le concept de "durée de rétention" d'une valeur de barème.

Une valeur de barème peut, comme on l'a vu, être conservée quelques temps après sa date de fin d'utilisation.

La durée pendant laquelle on désire encore garder une valeur, après sa fin d'utilisation, sera appelée "durée de rétention" de cette valeur.

Il y aura donc une durée de rétention par barème, cette durée étant valable pour toutes les valeurs de ce barème.

g) Le concept de valeur de barème "périmée"

Une valeur de barème est dite "périmée" si sa période de rétention est terminée. Elle est à supprimer du système.

h) Le concept de valeur de barème "historique"

Une valeur de barème est dite "historique" si celle-ci a été remplacée par une autre plus récente.

i) Le concept de "versions successives"

Soit une valeur de barème qui a été remplacée par une autre elle-même remplacée par une autre etc. Cet ensemble de valeurs, est appelé l'ensemble des "versions successives" d'une valeur.

j) Le concept de valeur de barème composée ou composante

Déjà introduit plus haut, on pourrait finalement définir ce concept par ce qui suit.

Une valeur de barème est dite composée si elle est la résultante d'une combinaison de N valeurs appartenant à N autres barèmes. Ces dernières valeurs simples, par contrainte, sont appelées "valeurs composantes".

2.3 Schéma Entités - Associations et ses contraintes.

a) Rappel théorique.

Avant de découvrir notre schéma, il est bon de procéder à un petit rappel théorique pour bien comprendre le formalisme utilisé.

Nous allons appliquer ce qu'on appelle le modèle "entité/association". Rappelons-nous donc très brièvement ce dont il s'agit. Pour plus d'informations, le lecteur pourra consulter l'oeuvre de F.BOBART et Y.PIGNEUR intitulé : "Conception des applications informatiques", parue aux éditions MASSON.

On appelle une entité, une chose concrète ou abstraite appartenant au réel perçu et à propos de laquelle on veut enregistrer des informations. Une entité n'existe en tant que telle que par rapport à un individu ou un groupe qui la considère comme un tout, lui confère une existence autonome et la distingue d'autres entités et de son environnement. Un type d'entité est la classe de toutes les entités possibles du réel perçu qui vérifie la définition constitutive du type.

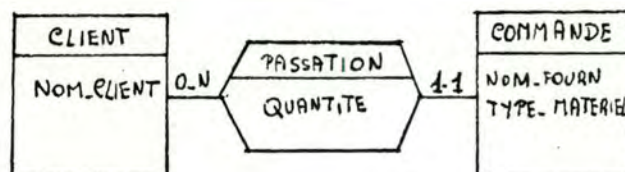
Une association est définie par une correspondance entre deux ou plusieurs entités (non nécessairement distinctes) où chacune assume un rôle donné. On appelle "type d'association", la classe de toutes les associations possibles du réel perçu qui vérifient la définition constitutive du type.

Les entités et les associations peuvent posséder des attributs c-à-d des caractéristiques ou des qualités qui leur sont liées.

Par convention, on représentera un type d'entité par un rectangle comportant un cartouche où figure le nom du T.E. On représentera un type d'association par un hexagone relié par des segments de droite aux rectangles qui représentent les T.E. sur lesquels est défini ce type d'association. Dans le cartouche supérieur de

l'hexagone, on indique le nom du T.A. Dans les cartouches inférieurs on y place les noms des attributs.

Exemple :



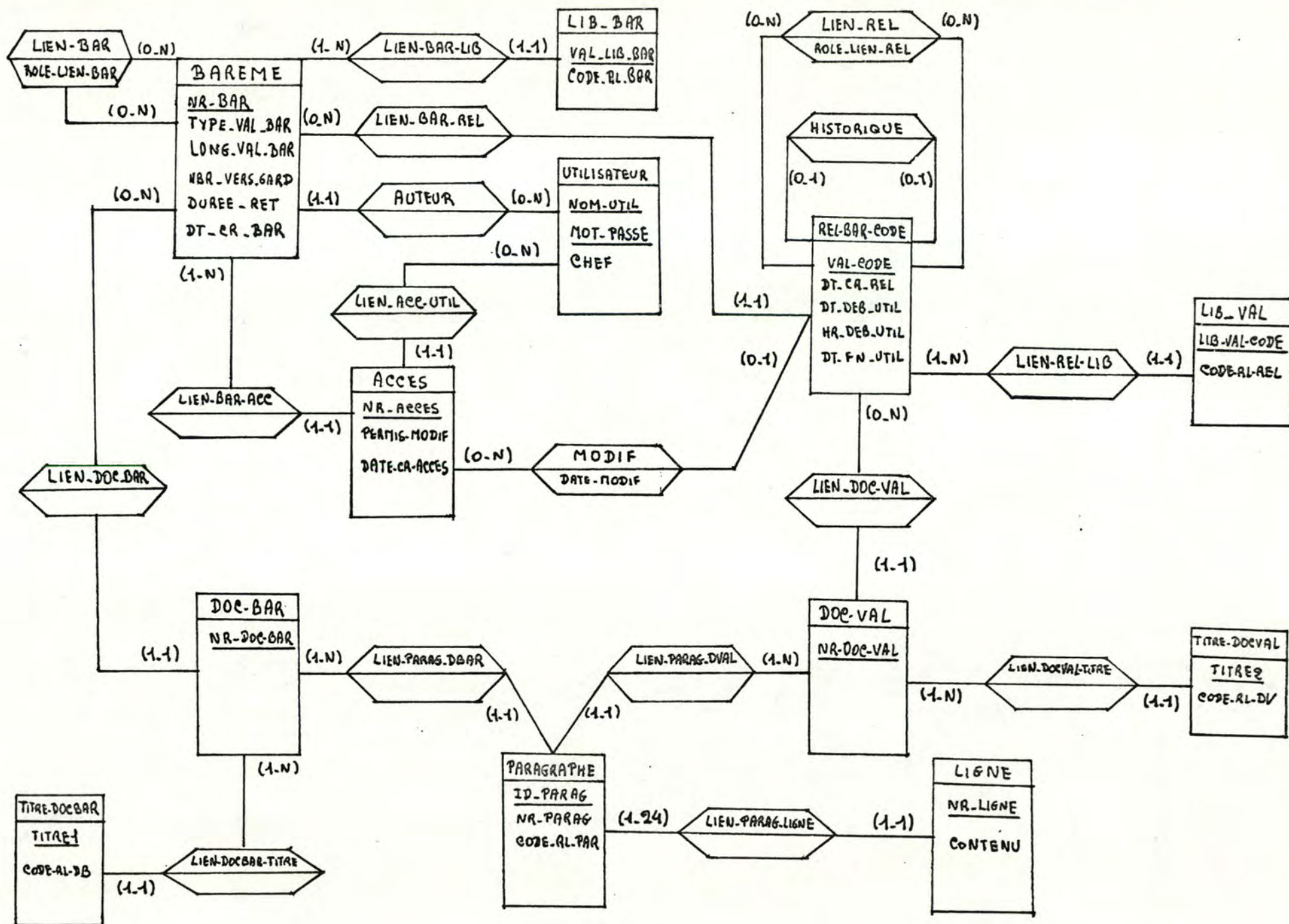
0-N et 1-1 sont, ce qu'on appelle, les connectivités du T.A. 0-N signifie qu'un client peut passer N commandes mais peut aussi ne pas en passer du tout.

Et pour 1-1 ? Le premier 1 signifie qu'une commande doit être reliée par le T.A. PASSATION à un client. Le deuxième 1 nous précise que cette commande doit être reliée à un et un seul client.

Malheureusement, on ne peut tout dire par ce formalisme. Il faut donc ajouter au schéma un certain nombre de contraintes que l'on appelle "contraintes d'intégrité".

Voici maintenant le schéma E/A qui résulte de cette analyse. Ensuite nous décrirons et définirons toutes les entités qui ont été identifiées. Puis suivront les associations qui relient ces entités et nous termineront par un exposé des différentes contraintes d'intégrité qui sont liées au schéma. Nous exposerons les types d'entités, d'associations, et les attributs du schéma selon la grille de description qui suit :

Type d'entité X	*	Type d'association Z
Définition	*	Définition
Identifiant	*	Entités reliées
	*	Connectivités
Attributs	*	Attributs
(Définition, valeurs	*	(Idem)
possibles, Forme cobol)	*	



c) Dictionnaire des types d'entités, des types d'associations et de

 et de leurs attributs.

c1) Type d'entité BAREME.

Définition: C'est un répertoire de valeurs distinctes créé

 pour les besoins de l'entreprise.

Identifiant: NR-BAR

Attributs:

NR-BAR

Déf: Numéro attribué à un barème lors de la création de
 celui-ci. Ce numéro permet d'identifier ce barème
 parmi les autres.

Val: Chaîne de caractères.

Forme cobol: X(6).

TYPE-VAL-BAR

Déf: Type des valeurs de code dans un barème.

Val: "AN" pour alphanumérique.

"AB" pour alphabétique.

"NU" pour numérique.

Forme cobol: X(2).

LONG-VAL-BAR

Déf: Longueur maximale des valeurs de code dans un barème
 particulier. C'est donc le nombre maximum de posi-
 tions que cette valeur peut prendre.

Val: de 01 à 06.

Forme cobol: 9(2).

NBR-VERS-GARD

Déf: Nombre maximum de versions successives à garder en historique concernant une valeur de barème.

Val: de 00 à 20.

Forme cobol: 9(2).

DUREE-RET

Déf: Durée pendant laquelle une valeur quelconque d'un barème doit être gardée après sa date de fin d'utilisation. Cette durée est exprimée en nombre de jours.

Val: de 0001 à 9999.

Forme cobol: 9(4).

DT-CR-BAR

Déf: Date à laquelle le barème en question a été créé.

Val: Dates compatibles avec le calendrier.

Exemple: "01/01/1986".

Forme cobol: X(10).

c2) Type d'entité REL-BAR-CODE.

Définition: Entité exprimant la participation d'une valeur

 de code dans un barème. L'ensemble des valeurs
 de code d'un barème forme le domaine des valeurs
 de code valides pour ce barème. A cette relation
 peut aussi être reliés un ou plusieurs libellés.

Identifiant: VAL-CODE et NR-BAR via LIEN-BAR-REL.

Attributs:

VAL-CODE

Déf: Valeur appartenant au code qui a été défini sur le barème en vue de représenter une information de manière plus compacte.

Val: chaîne de caractères.

Forme cobol: X(6).

DT-CR-REL

Déf: Date à laquelle une valeur de code a commencé à participer à un barème.

Val: Dates compatibles avec le calendrier.

Forme cobol: X(10).

DT-DEB-UTIL

Déf: Date de début d'utilisation d'une valeur de code dans sa participation à un barème.

Val: Dates compatibles avec le calendrier.

Forme cobol: X(10).

HR-DEB-UTIL

Déf: Heure de début d'utilisation d'une valeur de code dans sa participation à un barème.

Val: Toutes heures valides.

Exemple: "12H00".

Forme cobol: X(5).

DT-FN-UTIL

Déf: Date de fin d'utilisation d'une valeur de code dans sa participation à un barème.

Val: Dates compatibles avec le calendrier.

Forme cobol: X(10).

c3) Type d'entité LIB-BAR.

Définition: Libellé d'un barème dans un certain régime linguistique.

Identifiant: VAL-LIB-BAR.

Attributs:

VAL-LIB-BAR

Déf: Libellé pouvant être attribué lors de la création de ce barème.

Val: Chaîne de caractères.

Forme cobol: X(50).

CODE-RL-BAR

Déf: Code régime linguistique associé au libellé de ce barème.

Val: AN (anglais), FR (français), NR (néerlandais),
ou AL (allemand).

Forme cobol: X(2).

c4) Type d'entité LIB-VAL.

Définition: Libellé d'une valeur de code dans un certain régime linguistique.

Identifiant: LIB-VAL-CODE.

Attributs:

LIB-VAL-CODE

Déf: Libellé en question pouvant être attribué lors de la création d'une relation entre un barème et une valeur de code.

Val: Chaîne de caractères.

Forme cobol: X(50).

CODE-RL-REL

Déf: Code régime linguistique associé au libellé de cette valeur de code.

Val: AN (anglais) , FR (français) , NR (néerlandais),
ou AL (allemand).

Forme cobol: X(2).

c5) Type d'entité UTILISATEUR.

Définition: Toute personne susceptible de créer, consulter

et mettre à jour un barème.

Identifiant: NOM-UTIL et MOT-PASSE

Attributs:

NOM-UTIL

Déf: Nom de l'utilisateur qui a accès au système.

Val: Chaîne de caractères.

Forme cobol: X(10).

MOT-PASSE

Déf: Mot de passe de l'utilisateur qui a accès au système.

Val: Chaîne de caractères.

Forme cobol: X(10).

CHEF

Déf: Indicateur qui stipule si cet utilisateur est le chef du système ou pas.

Val: "O" ou "N".

Forme cobol: X(1).

c6) Type d'entité DOC-BAR.

Définition: Documentation portant sur un barème particulier.

Identifiant: NR-DOC-BAR.

Attributs:

NR-DOC-BAR

Déf: Numéro identifiant une documentation portant sur un barème particulier attribué lors de la création de cette documentation.

Val: de 001 à 999.

Forme cobol: 9(3).

c7) Type d'entité DOC-VAL

Définition: Documentation portant sur une valeur de code

 participant à un barème.

Identifiant: NR-DOC-VAL.

Attributs:

NR-DOC-VAL

Déf: Numéro identifiant une documentation portant sur une
 valeur participant à un barème. Ce numéro est attribué
 lors de la création de cette documentation.

Val: de 0001 à 9999.

Forme cobol: 9(4).

c8) Type d'entité PARAGRAPHE

Définition: Paragraphe d'une documentation portant sur un

 barème particulier ou sur une valeur de code
 participant à un barème.

Identifiant: ID-PARAG

Attributs:

ID-PARAG

Déf: Numéro identifiant un paragraphe parmi les autres,
 attribué lors de la création de ce paragraphe.

Val: de 000001 à 999999.

Forme cobol: 9(6).

NR-PARAG

Déf: Numéro d'ordre du paragraphe dans la documentation auquel il appartient.

Val: De 01 à 20.

Forme cobol: 9(2).

CODE-RL-PAR

Déf: Code régime linguistique associé au paragraphe.

Val: FR (français), AN (anglais), NR (néerlandais),
AL (allemand).

Forme cobol: X(2).

c9) Type d'entité LIGNE.

Définition: C'est un morceau de paragraphe ou plus précisément,

ment, une chaîne de 80 caractères au maximum.

Identifiant: NR-LIGNE et ID-PARAG via LIEN-PARAG-LIGNE.

Attributs:

NR-LIGNE

Déf: Numéro d'ordre de la ligne en question dans le paragraphe auquel elle est reliée.

Val: de 01 à 24.

Forme cobol: 9(2).

CONTENU

Déf: Ce sont les caractères qui constituent la ligne.

Val: chaîne de caractères.

Forme cobol: X(80).

c10) Type d'entité TITRE-DOC-BAR.

Définition: Titre d'une documentation portant sur un barème.

Il est exprimé dans un régime linguistique.

Identifiant: TITRE1.

Attributs:

TITRE1

Déf: Attribut contenant le titre en question.

Val: chaîne de caractères.

Forme cobol: X(30).

CODE-RL-DB

Déf: Code régime linguistique associé au titre en question

Val: FR (français), AN (anglais), NR (néerlandais),
AL (allemand).

Forme cobol: X(2).

c11) Type d'entité TITRE-DOC-VAL

Définition: Titre d'une documentation portant sur une valeur

de code. Il est exprimé dans un certain régime
linguistique.

Identifiant: TITRE2.

Attributs:

TITRE2

Déf: Attribut contenant le titre de la documentation.

Val: chaîne de caractères.

Forme cobol: X(30).

CODE-RL-DV

Déf: Code régime linguistique associé à ce titre.

Val: FR (français) , AN (anglais) , NR (néerlandais),
AL (allemand).

Forme cobol: X(2).

c12) Type d'entité ACCES.

Définition: Autorisation pour un utilisateur d'accéder à un

barème particulier. Bien entendu, le créateur
d'un barème possède d'office les droits d'accès
et de modification à son barème.

Identifiant: NR-ACCES

Attributs:

NR-ACCES

Déf: Numéro identifiant un droit d'accès parmi les autres
et attribué lors de la création de ce droit.

Val: de 001 à 999.

Forme cobol: 9(3).

PERMIS-MODIF

Déf: Code qui définit si oui ou non un utilisateur, ayant un droit d'accès sur un barème, peut le modifier.

Val: O (=oui), N (=non).

Forme cobol: X(1).

DATE-CR-ACCES

Déf: Date à laquelle ce droit d'accès a été défini.

Val: Dates valides par rapport au calendrier.

Forme cobol: X(10).

c13) Type d'association LIEN-BAR

Définition: Cette association signifie qu'un barème particulier est la combinaison d'autres barèmes existants.

Entités reliées: BAREME et BAREME*.

Connectivités: (0-N) pour BAREME. Un barème n'est pas nécessairement la combinaison d'autres barèmes.

(0-N) pour BAREME*. Un barème n'est pas nécessairement combiné avec d'autres barèmes.

Attribut:

ROLE-LIEN-BAR

Déf: Rôle joué par ce lien entre deux barèmes.

Val: Chaîne de caractères.

Forme cobol: X(50).

c14) Type d'association LIEN-REL.

Définition: Cette association nous indique qu'une valeur de

code participant à un barème est la combinaison d'autres valeurs de code participant à d'autres barèmes.

Entités reliées: REL-BAR-CODE et REL-BAR-CODE*.

Connectivités: (0-N) pour REL-BAR-CODE. Une valeur de code

ne fait pas nécessairement l'objet d'une telle combinaison.

(0-N) pour REL-BAR-CODE*. Une valeur de code n'est pas nécessairement combinée avec d'autres.

Attribut:

ROLE-LIEN-REL

Déf: Rôle joué par ce lien entre deux valeurs de code.

Val: Chaîne de caractères.

Forme cobol: X(80).

c15) Type d'association LIEN-BAR-REL.

Définition: Cette association nous indique qu'un barème est

mis en correspondance avec une série de relations, chacune d'elles, désignant une valeur de code valide pour ce barème et aussi une série de libellés liés à cette valeur de code.

Entités reliées: BAREME et REL-BAR-CODE.

Connectivités: (0-N) pour BAREME. Un barème peut exister

sans pour autant être relié à une telle relation. Exemple: lors de sa création.

(1-1) pour REL-BAR-CODE. Une telle relation ne peut porter que sur un et un seul barème.

c16) Type d'association HISTORIQUE.

Définition: Cette association relie une relation à une autre

 qui est considérée comme valeur historique de la première.

Entités reliées: REL-BAR-CODE et REL-BAR-CODE*.

Connectivités: (0-1) pour REL-BAR-CODE*. Car une valeur de

 code dans un barème ne possède pas nécessairement de valeur historique qu'elle a remplacée à une certaine date.
 (0-1) pour REL-BAR-CODE. Une valeur de code dans un barème ne possède pas nécessairement de valeur suivante dans le temps.

c17) Type d'association LIEN-BAR-LIB.

Définition: Cette association signifie qu'un barème quelcon-

 que possède un ou plusieurs libellés, chacun dans un régime linguistique particulier.

Entités reliées: BAREME et LIB-BAR.

Connectivités: (1-N) pour BAREME.

 (1-1) pour LIB-BAR. Un libellé se rapportera à un et un seul barème.

c18) Type d'association LIEN-REL-LIB.

Définition: Cette association signifie qu'une relation

 liée à une valeur de code, est associée à un ou

plusieurs libellés, chacun dans un régime linguistique déterminé.

Entités reliées: REL-BAR-CODE et LIB-VAL.

Connectivités: (1-N) pour REL-BAR-CODE.

(1-1) pour LIB-VAL.

c19) Type d'association AUTEUR.

Définition: Cette association permet de rattacher un barème

à son créateur.

Entités reliées: BAREME et UTILISATEUR.

Connectivités: (0-N) pour UTILISATEUR. Un utilisateur peut

ne pas être auteur mais peut l'être N fois.

(1-1) pour BAREME. Un barème a un et un seul

créateur.

c20) Type d'association LIEN-PARAG-LIGNE.

Définition: Cette association permet de relier un paragraphe

particulier aux lignes qui le composent.

Entités reliées: PARAGRAPHE et LIGNE.

Connectivités: (1-24) pour PARAGRAPHE. On considère qu' il y

a au moins une ligne et au plus vingt-quatre.

(1-1) pour LIGNE. Une ligne est reliée à un

seul paragraphe.

c21) Type d'association LIEN-DOC-BAR.

Définition: Association qui permet de relier un barème à ses

documentations.

Entités reliées: BAREME et DOC-BAR.

Connectivités: (0-N) pour BAREME. Un barème peut ne pas posséder de documentation.

(1-1) pour DOC-BAR. Une documentation concerne un et un seul barème.

c22) Type d'association LIEN-DOC-VAL.

Définition: Association qui permet de relier une valeur de code d'un barème à ses documentations.

Entités reliées: REL-BAR-CODE et DOC-VAL.

Connectivités: (0-N) pour REL-BAR-CODE. (Mêmes explications que pour le point d9).

(1-1) pour DOC-VAL. (Idem)

c23) Type d'association LIEN-PARAG-DBAR.

Définition: Cette association signifie qu'une documentation concernant un barème est subdivisée en un ou plusieurs paragraphes. Chaque paragraphe existe en autant de régimes linguistiques que l'on veut (4 au maximum).

Entités reliées: DOC-BAR et PARAGRAPHE.

Connectivités: (1-N) pour DOC-BAR. Une documentation possède au moins un paragraphe.

(1-1) pour PARAGRAPHE. Un paragraphe ne peut porter que sur une seule documentation.

c24) Type d'association LIEN-PARAG-DVAL.

Définition: Cette association signifie qu'une documentation

concernant une valeur de code d'un barème est subdivisée en un ou plusieurs paragraphes.
Chacun de ceux-ci existe en autant de régimes linguistiques que l'on veut (4 au maximum).

Entités reliées: DOC-VAL et PARAGRAPHE.

Connectivités: (1-N) pour DOC-VAL. Une documentation possède

au moins un paragraphe.

(1-1) pour PARAGRAPHE. Un paragraphe ne peut être lié qu'à une et une seule documentation.

c25) Type d'association LIEN-ACC-UTIL.

Définition: Association qui permet de lier à un utilisateur

tous les droits d'accès qu'il a obtenus sur les barèmes existant dans le système.

Entités reliées: UTILISATEUR et ACCES.

Connectivités: (0-N) pour UTILISATEUR. Un utilisateur peut

exister en n'ayant encore aucun accès.

(1-1) pour ACCES. Un droit d'accès à un barème ne peut être relié qu'à un utilisateur.

c26) Type d'association LIEN-BAR-ACC.

Définition: Association qui permet de relier à un barème

particulier, tous les droits d'accès qui ont été définis dessus.

Entités reliées: BAREME et ACCES.

Connectivités: (1-N) pour BAREME. Il existe au moins un

droit d'accès sur un barème, celui de son créateur.

(1-1) pour ACCES. Un droit d'accès ne porte que sur un et un seul barème.

c27) Type d'association LIEN-DOCBAR-TITRE.

Définition: Cette association permet de rattacher une documentation concernant un barème , au(x) titre(s) de cette documentation.

Entités reliées: DOC-BAR et TITRE-DOCBAR.

Connectivités: (1-N) pour DOC-BAR. une documentation possède un ou plusieurs titres dans des régimes linguistiques différents.

(1-1) pour TITRE-DOCBAR. Un titre porte sur une et une seule documentation.

c28) Type d'association LIEN-DOCVAL-TITRE.

Définition: Cette association permet de rattacher une documentation concernant une valeur de barème au(x) titre(s) de cette documentation.

Entités reliées: DOC-VAL et TITRE-DOCVAL.

Connectivités: (1-N) pour DOC-VAL. (Même explication que pour le T.A. précédant).

(1-1) pour TITRE-DOC-VAL. (Idem).

c29) Type d'association MODIF.

Définition: Cette association signifie qu'un utilisateur ayant un droit d'accès sur un barème avec modifications autorisées, a remplacé une valeur de ce barème. On retrouvera donc par cette asso-

ciation l'occurrence du T.E. REL-BAR-CODE qui a été modifiée.

Entités reliées: ACCES et REL-BAR-CODE.

Connectivités: (0-N) pour ACCES. Un utilisateur peut ne fai-

re aucune modification.

(0-1) pour REL-BAR-CODE. Une valeur modifiée

ne se rattache qu'à un seul accès. Par contre

une valeur peut ne pas être modifiée.

Attribut:

DATE-MODIF.

Déf: Date à laquelle la modification a eu lieu.

Val: Dates compatibles avec le calendrier.

Forme cobol: X(10).

d) Les contraintes d'intégrité du schéma.

-
- d1) Dans une occurrence du T.E. REL-BAR-CODE, la valeur de l'attribut DT-DEB-UTIL doit toujours être inférieure ou égale à la valeur de l'attribut DT-FIN-UTIL.
 - d2) Dans une occurrence du T.E. REL-BAR-CODE, la valeur de l'attribut DT-CR-UTIL doit toujours être inférieure ou égale à la valeur de l'attribut DT-DEB-UTIL. Si ces deux attributs ont la même valeur, alors la valeur de l'attribut HR-DEB-UTIL doit être supérieure à l'heure du système au moment de la création de cette occurrence.
 - d3) Par la participation d'une occurrence du T.E. REL-BAR-CODE dans une occurrence du T.A. MODIF, on peut donc trouver l'occurrence visée du T.E. ACCES et par celle-ci, remonter via LIEN-BAR-ACC à l'occurrence du T.E. BAREME visée par cette

dernière association. Si on part de la même occurrence de REL-BAR-CODE et qu'on remonte via LIEN-BAR-REL, on doit retrouver la même occurrence de BAREME.

d4) La période de validité (cfr définition des concepts) d'une valeur de barème composé doit être strictement comprise dans l'intersection des périodes de validité de ses valeurs composantes.

d5) Toute occurrence du T.E. REL-BAR-CODE qui participe à une occurrence du T.A. HISTORIQUE, doit nécessairement participer à une occurrence du T.A. MODIF.

d6) La durée de rétention d'un barème composé doit être celle de ses barèmes simples qui est minimale. Cette contrainte est nécessaire pour assurer l'intégrité du système au moment de son nettoyage. En effet, si une valeur simple composante doit être supprimée alors, les valeurs composées où elle intervient doivent l'être au moins avant elle.

Ces deux dernières contraintes auraient été difficiles à exprimer en terme d'occurrences du schéma.

d7) Toutes les valeurs de l'attribut VAL-CODE des occurrences du T.E. REL-BAR-CODE liées à la même occurrence du T.E. BAREME via le T.A. LIEN-BAR-REL doivent:

- être de type alphanumérique si la valeur de l'attribut TYPE-VAL est "AN",
- être de type alphabétique si la valeur de l'attribut TYPE-VAL est "AB",
- être de type numérique si la valeur de l'attribut TYPE-VAL est "NU".

- d8) Toutes les valeurs de l'attribut VAL-CODE des occurrences du T.E. REL-BAR-CODE liées à la même occurrence du T.E. BAREME via le T.A. LIEN-BAR-REL, doivent avoir un nombre de caractères qui n'excède pas la valeur de l'attribut LONG-VAL de cette occurrence de BAREME.
- d9) Toutes les occurrences du T.E REL-BAR-CODE dont la valeur de l'attribut DT-FIN-UTIL est inférieure à la date du système et dont la durée de rétention (liée à l'occurrence du T.E. BAREME via une occurrence du T.A. LIEN-BAR-REL) est écoulée, doivent être supprimées du système.
- d10) Si un barème est composé de trois autres barèmes simples X,Y et Z, une valeur de ce barème composé doit être liée à trois autres valeurs qui appartiennent respectivement et nécessairement à ces trois barèmes simples.
- d11) Soit une occurrence de BAREME dont la valeur de NBR-VERS-GARD vaut 1 et soit une valeur A de ce barème qui est d'actualité, alors cette valeur A peut posséder une valeur historique B et cette dernière ne peut pas elle-même en posséder une via HISTORIQUE. Pour cela, NBR-VERS-GARD aurait dû valoir 2. Donc, dans un barème, le nombre d'occurrences de REL-BAR-CODE enchaînées via HISTORIQUE, à une occurrence de REL-BAR-CODE d'actualité, ne peut excéder la valeur de l'attribut NBR-VERS-GARD de ce barème. De plus, les valeurs qui font partie de cette "chaîne" ne doivent plus être d'actualité. Lors d'une modification, on met donc fin à la période d'utilisation de la valeur modifiée en changeant la valeur de l'attribut DATE-FN-UTIL.

- d12) Une occurrence de T.E. ACCES ne peut participer à une occurrence du T.A. MODIF que si la valeur de l'attribut PERMIS-MODIF est "0". Par ceci, on exprime que l'on ne peut modifier une valeur de barème si l'on ne possède pas le droit d'accès et de modification sur ce barème.
- d13) La participation d'une occurrence du T.E. UTILISATEUR dans une occurrence du T.A AUTEUR inclut sa participation dans une occurrence du T.A LIEN-ACC-UTIL. De plus, l'occurrence du T.E. ACCES, reliée à cette occurrence d'UTILISATEUR via LIEN-ACC-UTIL, doit être reliée à l'occurrence du T.A BAREME, via LIEN-BAR-UTIL, qui participe à l'occurrence du T.E. AUTEUR pour cette occurrence d'UTILISATEUR.
- d14) La participation d'une occurrence du T.E PARAGRAPHE dans une occurrence du T.A. LIEN-PARAG-DBAR exclut sa participation dans une occurrence du T.A. LIEN-PARAG-DVAL et vice versa.
- d15) La participation d'une occurrence du T.E. DOC-BAR dans une occurrence du T.A. LIEN-DOCBAR-TITRE inclut sa participation dans au moins une occurrence du T.A. LIEN-PARAG-DBAR dont la valeur de l'attribut CODE-RL-DB de l'occurrence de TITRE-DOCBAR est la même que celle de l'attribut CODE-RL-PAR de l'occurrence de PARAGRAPHE.
- d16) La participation d'une occurrence du T.E. DOC-VAL dans une occurrence du T.A. LIEN-DOCVAL-TITRE inclut sa participation dans au moins une occurrence du T.A. LIEN-PARAG-DVAL dont la valeur de l'attribut CODE-RL-DV de l'occurrence de TITRE-DOCVAL est la même que celle de l'attribut CODE-RL-PAR de l'occurrence de PARAGRAPHE.

d17) Un barème composé doit l'être de barèmes simples. Une valeur de ce barème est donc reliée à des valeurs simples d'actualité au moment de la création de cette valeur. Cette contrainte porte sur tous les REL-BAR-CODE qui doivent être reliés à d'autres occurrences de ce type via LIEN-REL.

d18) Dans le système, il ne peut y avoir qu'une seule occurrence du T.E UTILISATEUR dont la valeur de l'attribut CHEF est 0.

2.4 Description des traitements.

Voici maintenant la spécification des traitements que l'on a choisis de réaliser dans notre prototype. Nous commencerons par les traitements interactifs pour suivre par les traitements par programme. Ceux-ci seront spécifiés sous la forme : "Entrées - Sorties - Objectif". Remarquons que pour la partie interactive, les entrées/sorties concernent l'écran.

a) Les traitements interactifs.

%%%

1) La création de barème.

Entrées :

- Un identificateur de barème;
- Les noms ou libellés de ce barème (et régimes linguistiques associés).
- La longueur maximale de ses valeurs de code;
- Le type de ses valeurs de code;
- La date de création de ce barème;
- La durée de rétention de ses valeurs;

- Le nombre de versions successives d'une valeur que l'on peut garder en historique;
- Et si le barème est composé, un libellé par barèmes composants.

Sorties :

- En cas d'échec, un message avertit l'utilisateur.
- En cas de succès, l'en-tête d'un nouveau barème est créé dans le système. Cet en-tête comprend les renseignements que l'on a déterminé en entrée.

Objectif :

Le but de ce traitement est de créer à partir de l'écran l'en-tête d'un nouveau barème dans le système. Cette création est soumise aux contraintes qui suivent :

- la date de création du barème est en fait la date du jour;
- l'utilisateur qui crée un barème composé doit avoir accès aux barèmes qui doivent composer ce barème;
- la création de barèmes composés se fera s'il existe déjà des barèmes dans le système sinon la détermination de barèmes composants est impossible.
- (+ 2.3 d16 et d17).

Cette création peut se voir interrompue pour trois raisons :

- il n'y a pas de barèmes dans le système (cfr ci-dessus);
- l'utilisateur a désigné un barème composant auquel il n'avait pas accès.
- l'utilisateur a échoué trois fois dans la détermination d'un barème composant. Les causes d'un tel échec sont les suivantes :
 - le barème désigné était composé;

- le barème désigné l'avait déjà été;
- le barème désigné n'existait pas;

2) La création d'une valeur de barème.

Entrées :

- le libellé du barème où il faut créer la valeur;
- et pour la valeur de barème à créer :
- une valeur de code;
- les libellés associés à cette valeur (+ régimes linguistiques);
- une date de création;
- une date et une heure de début d'utilisation;
- une date de fin d'utilisation;
- si le barème est composé, un ensemble de valeurs de code.

Sorties:

- En cas d'échec, un message doit avertir l'utilisateur.
- En cas de succès, une nouvelle valeur de barème est créée dans le système. Celle-ci comprend tous les renseignements lus en entrée.

Objectif:

Après détermination du barème sur lequel il porte, ce traitement doit créer une nouvelle valeur de barème dans le système.

Pour cela, il doit satisfaire les contraintes suivantes :

- la valeur de code ne doit pas déjà exister dans ce barème;
- la date de création est en fait la date du système;
- (+ 2.3 d1, d2, d4, d17)

Cette création peut se voir interrompue par le fait que l'utilisateur a échoué trois fois dans la détermination d'une valeur de code composante. Les motifs d'un tel échec sont :

- la valeur de code n'existe pas dans le barème composant;
- la valeur de code existe mais n'est pas d'actualité.

La création peut aussi ne pas avoir lieu pour les raisons suivantes:

- il n'existe aucun barème dans le système;
- le barème désigné est composé et un de ses barèmes composants ne contient pas de valeur;
- l'utilisateur n'a pas accès au barème désigné;
- le barème désigné n'existe pas dans le système.

3) L'obtention des libellés d'une valeur de code.

Entrées :

- le libellé du barème objet de la recherche;
- la valeur de code dont on désire les renseignements.

Sorties :

En cas de succès, on obtient à l'écran,

- la valeur de code,
- le(s) libellé(s) de cette valeur de code,
- la date de création de cette valeur,
- la date et l'heure de début d'utilisation,
- la date de fin d'utilisation de la valeur.

En cas d'échec, un message est envoyé à l'écran.

Objectif :

Après avoir déterminé le barème objet de la recherche puis la valeur de code dont on désire avoir les renseignements, le but de ce traitement est de rechercher dans le système les libellés de cette valeur de code ainsi que les dates et heures qui y sont liées. Si la recherche est fructueuse, les résultats seront affichés à l'écran, sinon un message préviendra l'utilisateur que la valeur de code donnée n'existe pas dans ce barème. La recherche peut ne pas avoir lieu pour les raisons suivantes :

- il n'existe aucun barème dans le système;
- l'utilisateur n'a pas accès au barème désigné;
- le barème désigné n'existe pas dans le système;
- le barème désigné ne contient aucune valeur.

4) L'affichage de barème.

Entrée : le libellé du barème que l'on veut afficher.

Sorties :

- En cas d'échec, un message est envoyé à l'écran.
- En cas de succès, on obtient à l'écran,
 - * l'en-tête du barème demandé (composition cfr trait. n°1)
 - * si le barème demandé est composé, au choix, on peut voir l'en-tête des barèmes composants
 - * toutes les lignes qui sont d'actualité dans ce barème
(Présentation sous forme de tableau de valeurs de code)

Objectif :

Après avoir déterminé le barème que l'on veut afficher, l'objet de ce traitement est de consulter à l'écran l'en-tête de ce

barème, au choix, les en-têtes des barèmes composants (s'il y en a) et les tableaux de valeurs d'actualité qui constituent ce barème (s'il existe encore de telles valeurs). Ce traitement peut échouer éventuellement pour les raisons suivantes :

- il n'existe aucun barème dans le système;
- le barème désigné n'existe pas;
- le barème existe mais l'utilisateur n'y a pas accès.

5) L'obtention de la composition d'un barème.

Entrée : Le libellé du barème dont on désire la composition.

Sorties :

- En cas d'échec, un message est envoyé à l'écran;
- En cas de succès,
 - * si le barème est simple, on obtient à l'écran les libellés des barèmes qui l'admettent comme composant;
 - * si le barème est composé, on obtient à l'écran les libellés de ses barèmes composants.

Objectif :

Après avoir déterminé le barème dont on désire la composition, si celui-ci est simple, alors l'objet de ce traitement est de rechercher dans le système et d'afficher à l'écran les libellés des barèmes qui l'admettent comme composant; par contre si notre barème est composé, il s'agit alors des libellés des barèmes qui le composent. Ce traitement peut être interrompu pour les mêmes raisons que le traitements précédents.

6) L'obtention des noms de tous les barèmes.

Entrée : /

Sorties :

- On obtient à l'écran la liste de tous les libellés des barèmes du système.
- ou alors un message prévenant qu'il n'existe encore aucun barème dans le système.

Objectif :

Le but de ce traitement est de présenter à l'utilisateur la liste de tous les libellés des barèmes qui existent dans le système. Si la liste est vide, un message préviendra l'utilisateur. Ce traitement peut être interrompu par la commande "S" (=stop).

7) La recherche selon critère.

Entrées :

- le libellé du barème objet de la recherche;
- le critère de recherche composé d'un certain nombre de couples (Identificateur de barème, valeur de code composante).

Sorties :

- En cas d'échec, un message est envoyé à l'écran.
- En cas de succès, on obtient à l'écran un ou plusieurs tableaux comportant toutes les lignes qui sont d'actualité dans ce barème et qui contiennent toutes les valeurs de code composantes spécifiées dans le critère donné en entrée.

Objectif :

Ce traitement commence par identifier le barème objet de la recherche et le critère de cette recherche. Ce critère est valide si chaque identificateur de barème est distinct des autres, s'il est celui d'un des barèmes composants, et que la valeur de code associée existe dans ce barème. Ceci fait, le but de ce traitement est de voir à l'écran toutes lignes de ce barème qui sont d'actualité et qui contiennent les valeurs de code composantes contenues dans le critère de recherche.

La recherche peut encore ne pas avoir lieu pour les raisons suivantes :

- il n'existe aucun barème dans le système;
- le barème désigné n'existe pas dans le système;
- le barème existe mais l'utilisateur n'y a pas accès;
- le barème désigné est simple;
- le barème désigné ne contient aucune valeur;

(Remarque : ce cas masque le fait qu'un barème composant peut ne pas avoir de valeur. Si c'est le cas, le barème composé ne contient pas de valeur non plus et on ne pourra pas en créer tant que ses barèmes composants ne contiennent pas tous des valeurs)

- l'utilisateur a échoué trois fois dans la détermination d'une valeur de code composante "partie" du critère de recherche. Les causes de ces échecs sont les suivantes :

- * la valeur de code entrée n'existe pas dans le barème composant annoncé à l'écran;
- * ou elle n'est pas d'actualité.

8) L'obtention des accès autorisés pour un utilisateur.

Entrées : /

Sorties :

- l'utilisateur obtient à l'écran la liste de tous les libellés des barèmes du système auxquels il a accès,
- si cette liste est vide, le système le prévient par un message qu'il n'a encore aucun accès dans le système.

Objectif :

Le but de ce traitement est de sortir à l'écran la liste de tous les libellés des barèmes auxquels l'utilisateur, qui s'est introduit dans le système, a accès. Si cet utilisateur ne possède aucun accès, il est averti par un message.

Remarque : Ce dernier cas masque peut-être le fait qu'il n'existe aucun barème dans le système, cela entraînant le fait qu'il n'existe forcément aucun accès à des barèmes dans le système.

Ce traitement peut être interrompu par la commande "S" (=stop).

9) Le nettoyage du système.

Entrées : /

Sorties :

- une liste de libellés des barèmes (+ identificateur associé) des barèmes qui ont été supprimés du système.

Objectif :

L'objet de ce traitement est de passer en revue tout le système afin d'en supprimer toutes les valeurs de barème qui sont périmées (cfr 2.2.g déf. des concepts). Cette opération devra se faire en commençant par les barèmes composés car rappelons que si une valeur de code simple est périmée, toutes les lignes de barèmes composés où elle intervient le sont aussi et doivent être supprimées avant elle. Cela fait, il reste à répéter l'opération pour les barèmes simples, et à supprimer du système les barèmes qui ne contiennent plus de valeur.

Remarquons qu'un utilisateur ayant créé un barème sans y mettre de valeur avant l'opération de nettoyage, se verra supprimer son barème. De toute manière, ces utilisateurs doivent être au courant de ce fait qui n'a pas de conséquences graves.

Ceci constitue le cahier des charges de la partie interactive que l'on a choisi de réaliser. Par manque de temps, on a été contraint de ne pas réaliser la distribution de droits d'accès et de modifications relative à la création de barème. Pour cette raison, cette distribution n'est pas apparente dans la spécification. Il en résulte donc un sous-système pour la gestion de la sécurité qui a pour but de prouver qu'une gestion de ce type, c-à-d comme on l'a prévue, est effectivement possible.

b) Les traitements par programmes.

XXXXXXXXXXXXXXXXXXXXXXXXXXXX

Par ceci, on entend que l'on déclenchera les traitements qui suivent grâce à des appels par programmes. Après discussion, deux traitements à déclencher de cette façon ont pu être identifiés. Il s'agit en fait des traitements n° 3 et n° 7 de la partie interactive. Mais ici, les entrées sont des informations que le programme appelant transmet aux traitements en question, et les sorties sont des informations que notre traitement renvoie au programme qui l'a appelé.

Voici donc ci-dessous la spécification des deux traitements déterminés, ce qui termine notre analyse fonctionnelle.

1) L'obtention des libellés d'une valeur de code.

Entrées :

- Le nom de l'utilisateur qui désire accéder au système;
- son mot de passe;
- le libellé du barème objet de l'accès;
- la valeur de code dont on désire les renseignements.

Sorties :

- En cas de succès,
 - * la valeur de code;
 - * le(s) libellé(s) de cette valeur de code;
 - * la date de création de la valeur;
 - * la date et l'heure de début d'utilisation;
 - * la date de fin d'utilisation.

- En cas d'échec un numéro d'erreur est renvoyé. En voici le code avec sa signification :

- 0 pas d'erreur;
- 1 utilisateur pas reconnu;
- 2 barème inexistant;
- 3 l'utilisateur n'a pas d'accès à ce barème;
- 4 il n'y a aucune valeur dans ce barème;
- 11 la valeur de code n'existe pas;
- 12 il n'y a pas de barème dans le système.

Objectif :

Le but de ce traitement est de renvoyer au programme appelant les renseignements (cfr sorties) qui correspondent à la valeur de code et au barème spécifié en entrée. On obtiendra effectivement ceux-ci si aucune des six erreurs spécifiées ci-dessus ne se produit.

2) La recherche selon critère.

Entrées :

-
- Le nom de l'utilisateur qui désire accéder au système;
 - son mot de passe;
 - le libellé du barème objet de la recherche;
 - le critère de recherche composé d'un certain nombre de couples du type: (identificateur de barème, valeur de code composante).

Sorties :

- Toutes les lignes de ce barème qui sont d'actualité et qui contiennent toutes les valeurs de code composantes du critère de recherche,
- ou un numéro d'erreur est renvoyé. En voici le détail :
 - de 0 à 4 : idem que le traitement précédent;
 - 5 recherche interdite sur les barèmes simples;
 - 6 mauvais barème composant;
 - 7 valeur composante pas encore d'actualité;
 - 8 une valeur composante n'est plus d'actualité;
 - 9 une valeur composante est inexistante;
 - 10 aucun résultat
 - 12 idem que le traitement précédent.

Objectif :

Ce traitement a pour but de fournir au programme appelant toutes les lignes du barème spécifié en entrée qui contiennent toutes les valeurs de code contenue dans le critère de recherche donné. On les obtiendra effectivement si aucune des onzes erreurs spécifiées ci-dessus ne se produit.

C H A P I T R E I I I

Analyse d'implémentation

Dans ce chapitre, on trouvera la réalisation du prototype. Elle se fera en langage Cobol sur une base de données de type CODASYL. Ce chapitre comporte donc la description d'un schéma compatible CODASYL à partir du schéma E/A développé, et d'une architecture logicielle de programmation relative à notre système.

3.1 Schéma de la base de données.

a) Formation du schéma d'accès à partir du schéma E/A.

Tout d'abord, il est bon de rappeler les règles générales de transformation d'un schéma E/A en un schéma de base de données. On utilisera ici les cinq transformations qui suivent :

T1 / A chaque type d'entité est associé un type d'article.

T2 / A chaque attribut d'un type d'entité est associé un item d'un type d'article.

T3 / A chaque type d'association binaire sans attribut est associé un type de chemin avec inverse.

T4 / A chaque type d'association binaire avec attribut est associé un type d'article, des items et deux types de chemins avec inverses.

T5 / On utilisera aussi ce qu'on appelle la "rotation". c'est une transformation qui permet d'éliminer un type de chemin entre deux types d'articles.

Exemple :



Le "propriétaire" ou la "tête" du type de chemin APPARTENANCE est le type d'article DEPARTEMENT, et les membres sont les articles de type "EMPLOYE". Si on veut éliminer ce type de chemin, on effectue ce qu'on appelle une rotation du type de chemin APPARTENANCE vers la clé NR-DEP du type d'article DEPARTEMENT. En fait, cela signifie que l'on supprime ce type de chemin et qu'on ajoute un nouvel item NR-DEP-EMP (qui est le numéro du département auquel cet employé appartient) au type d'article EMPLOYE. Cette transformation implique qu'il faudra toujours s'assurer à la création d'un article EMPLOYE que la valeur de son item NR-DEP-EMP est bien un des numéros de département NR-DEP existants.

Après transformation, il en résulte ce schéma :

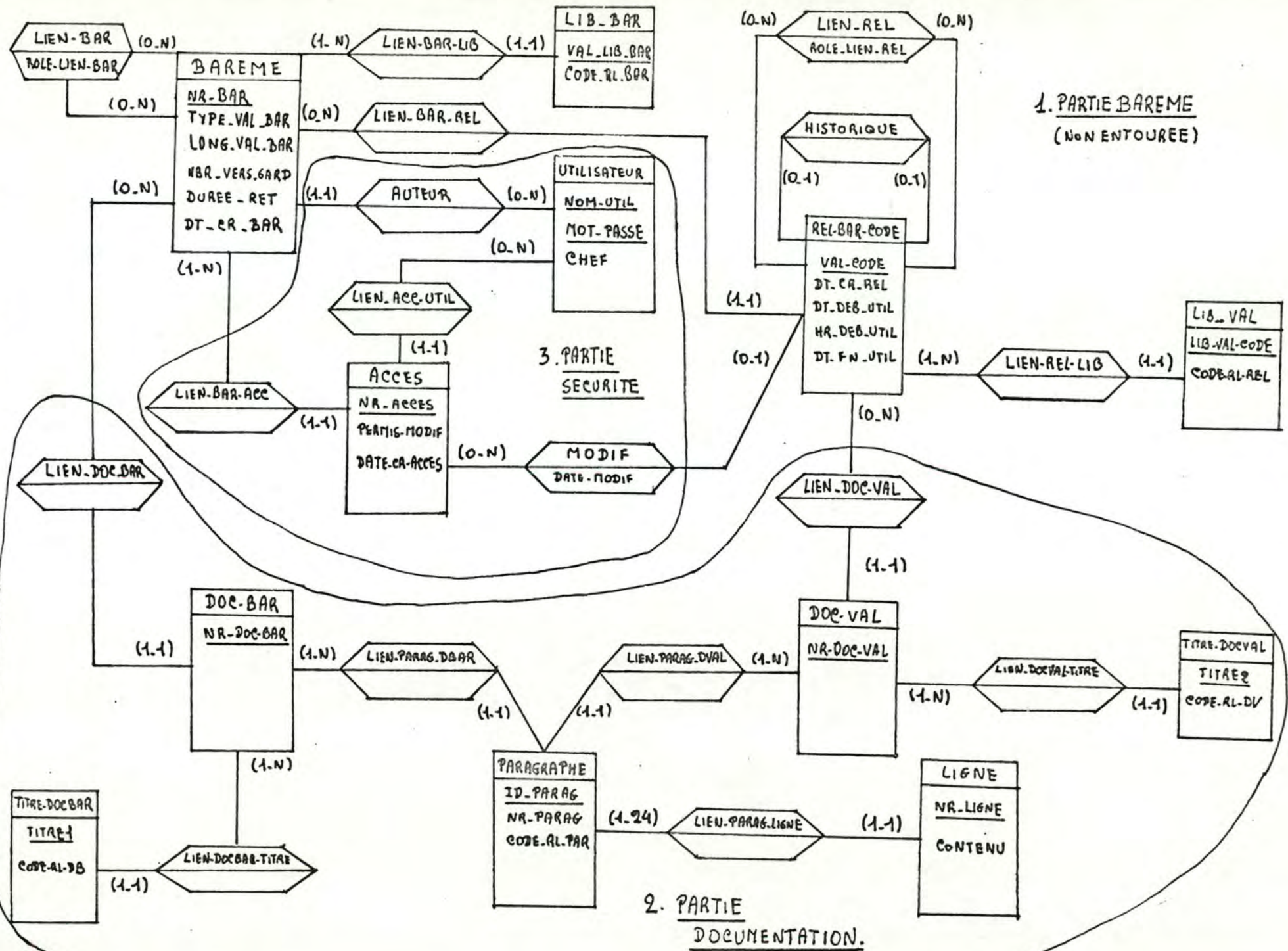


avec $\text{NR-DEP-EMP} (: \text{EMPLOYE}) = \text{NR-DEP} (: \text{DEPARTEMENT})$.

Ces transformations sont issues du document intitulé "Cadre de référence pour la conception de bases de données", écrit par J.L. HAINAUT au mois de décembre 1983, et du cours intitulé "Systèmes de gestion de bases de données", donné^{en} première licence et maîtrise en informatique (FUNDP), et issu du même auteur.

Afin de bien comprendre les transformations effectuées, proposons maintenant au lecteur de parcourir l'ensemble de celles-ci tout en ayant un oeil en regard sur le schéma E/A joint à la page 59.

Voici donc le schéma E/A origine de toutes les transformations :



Procédons donc aux transformations adéquates qui permettront d'obtenir un schéma d'accès CODASYL. Remarquons que lors des transformations, le nom de certains items par rapport à leurs attributs correspondants ont été volontairement changés. Ces noms étant significatifs, on associera facilement les uns aux autres.

Par T1 et T2, au type d'entité BAREME (et ses attributs) correspond le type d'article BAREME (et ses items) dont la clé d'accès identifiante est NR-BAR.

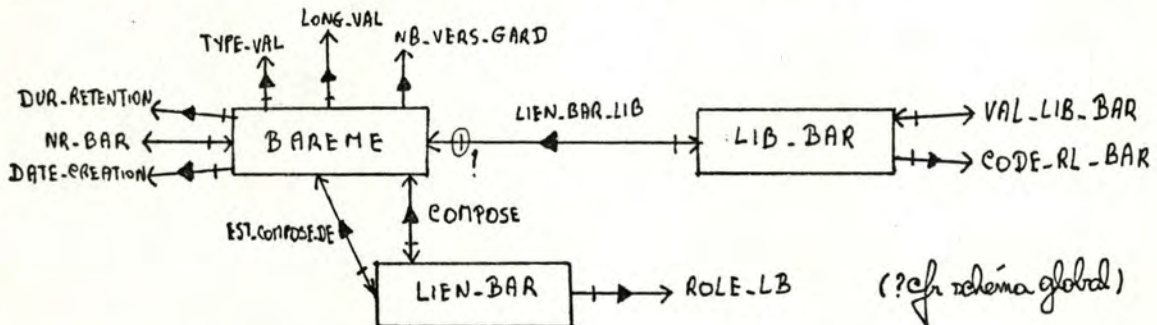
Par T1 et T2, au type d'entité LIB-BAR (et ses attributs), correspond le type d'article LIB-BAR (et ses items) avec une clé d'accès identifiante VAL-LIB-BAR.

Selon T4, au type d'association LIEN-BAR (et son attribut), correspond le type d'article LIEN-BAR (et son item) et deux types de chemins qu'on appellera "compose" et "est compose de " entre les types d'article BAREME et LIEN-BAR. L'insertion d'un article supplémentaire est non seulement nécessaire à cause de l'attribut lié au type d'association mais aussi parce que la norme CODASYL n'admet pas les types de chemins récursifs.

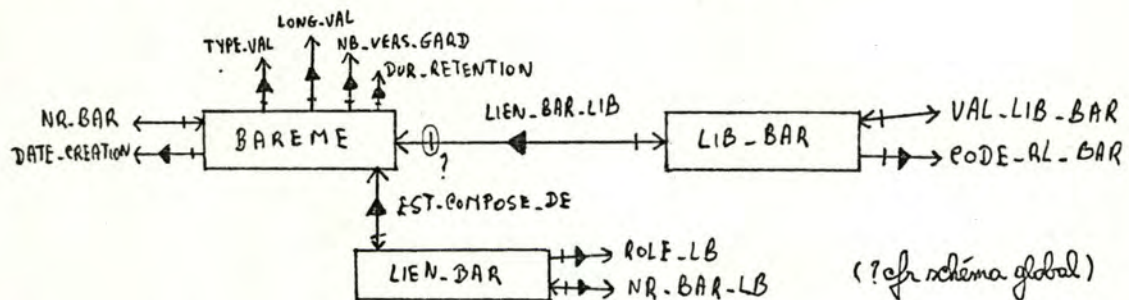
Cette transformation respecte la sémantique du schéma. A partir de cette solution, afin de ne pas multiplier les occurrences de types de chemin que la base de données doit gérer, on a choisi de supprimer un de ces deux types de chemins. On le fait en effectuant une rotation du type de chemin "compose" vers la clé de l'article barème visé. Pratiquement, il s'agit d'ajouter un item NR-BAR-LB au type d'article LIEN-BAR afin d'y mémoriser cette clé. Ce genre de changement inclut le fait qu'il faudra toujours veiller à ce que la valeur de cet item soit égale à une des valeurs des items NR-BAR (du type d'article BAREME) existants.

Selon T3, au type d'association LIEN-BAR-LIB correspond le type de chemin LIEN-BAR-LIB entre les types d'articles BAREME et LIB-BAR.

Ci-dessous se trouve donc la solution de "base" :



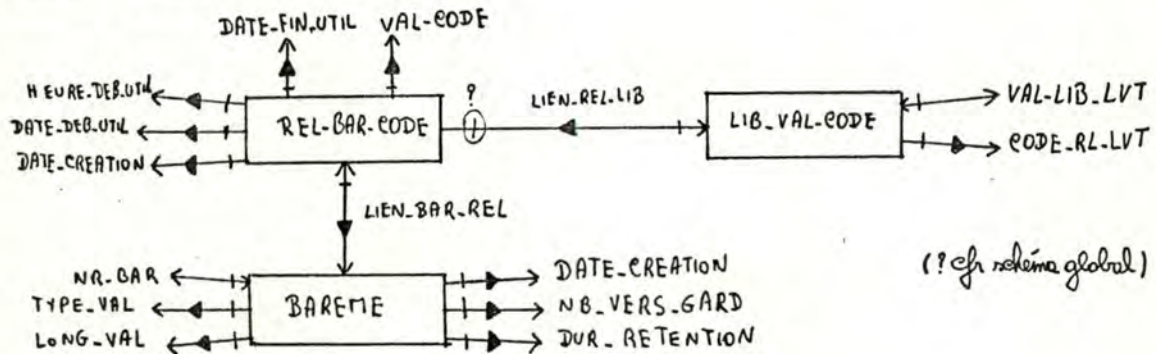
Voici la solution qui a été implémentée :



Par T1 et T2, au type d'entité REL-BAR-CODE (et ses attributs) correspond le type d'article REL-BAR-CODE (et ses items); et au type d'entité LIB-VAL (et ses attributs), correspond le type d'article LIB-VAL-CODE (et ses items).

Par T3, au type d'association LIEN-REL-LIB correspond un type de chemin entre les types d'articles REL-BAR-CODE et LIB-VAL-CODE; et au type d'association LIEN-BAR-REL correspond le type de chemin LIEN-BAR-REL entre les types d'articles BAREME et REL-BAR-CODE.

On obtient donc :

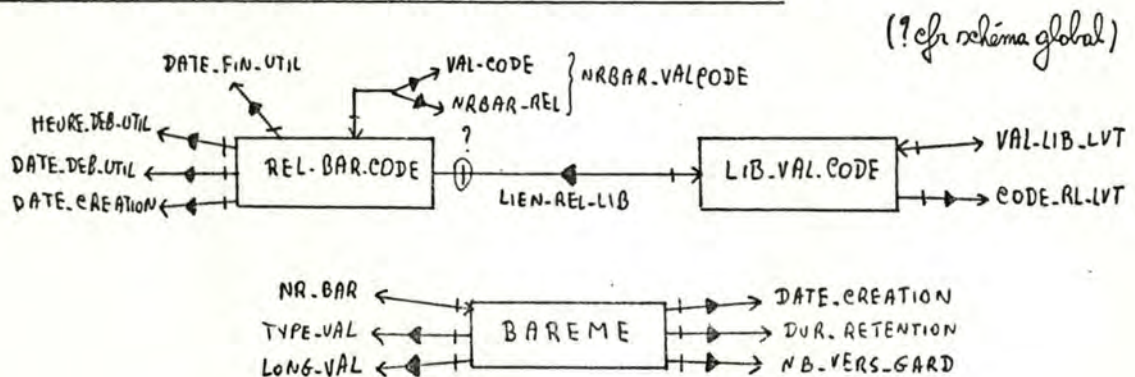


Voyons comment on peut améliorer ce schéma !

Comme un barème peut comprendre de nombreuses valeurs, les chemins de type LIEN-BAR-REL dont la tête est BAREME et les membres REL-BAR-CODE risquent d'être très longs et lourds à gérer. C'est pourquoi il est préférable d'éliminer ce type de chemin en effectuant une rotation de ce type de chemin vers la clé NR-BAR de l'article BAREME, en ajoutant un nouvel item NRBAR-REL au type d'article REL-BAR-CODE. Les valeurs de cet item devront bien entendu appartenir à l'ensemble des clés NR-BAR existantes.

Le type d'article REL-BAR-CODE possède maintenant une clé d'accès identifiante constituée des deux items NRBAR-REL et VAL-CODE. Ainsi possédant un numéro de barème et une valeur de code, on pourra accéder directement à l'article REL-BAR-CODE qui leur correspond. On appelle cette clé : "NRBAR-VALCODE".

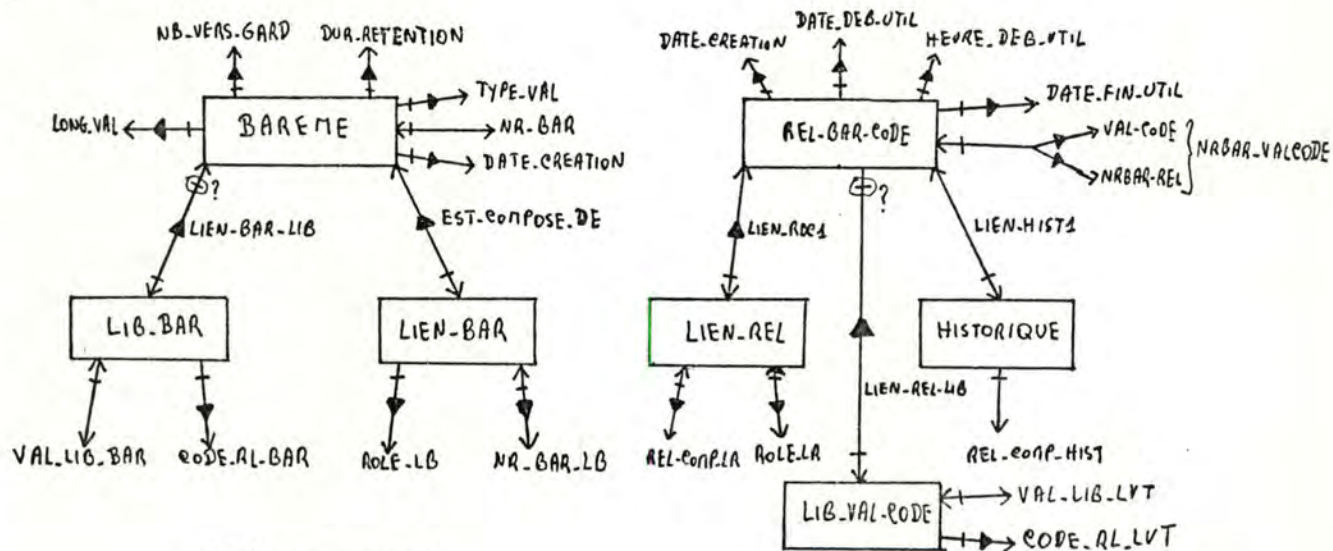
Le schéma modifié se représente donc ainsi :



Par T4, au type d'association LIEN-REL (et son attribut) correspond un type d'article LIEN-REL (et son item) et deux type de chemin "lien-rbc1" et "lien-rbc2". Vu le très grand nombre d'articles REL-BAR-CODE, on supprimera un des deux types de chemins pour alléger la gestion des chemins dans la base. On effectuera donc une rotation du type de chemin "lien-rbc2" vers la clé d'accès identifiante NRBAR-VALCODE de REL-BAR-CODE créant un nouvel item REL-COMP-LR pour le type d'article LIEN-REL. Une valeur quelconque de ce nouvel item devra toujours appartenir à l'ensemble de ces clés identifiantes NRBAR-VALCODE.

Par le fait que CODASYL n'admet pas les types de chemin récursif, au type d'association HISTORIQUE doit correspondre un type d'article HISTORIQUE et deux types de chemins "lien-hist1" et "lien-hist2". Pour la même raison que ci-dessus, on supprime par rotation le chemin "lien-hist2", en ajoutant un nouvel item REL-COMP-HIST au type d'article HISTORIQUE. De même, une valeur de cet item devra appartenir à l'ensemble des clés identifiantes NRBAR-VALCODE de REL-BAR-CODE.

On obtiendra finalement pour la partie du schéma qui concerne les barèmes :

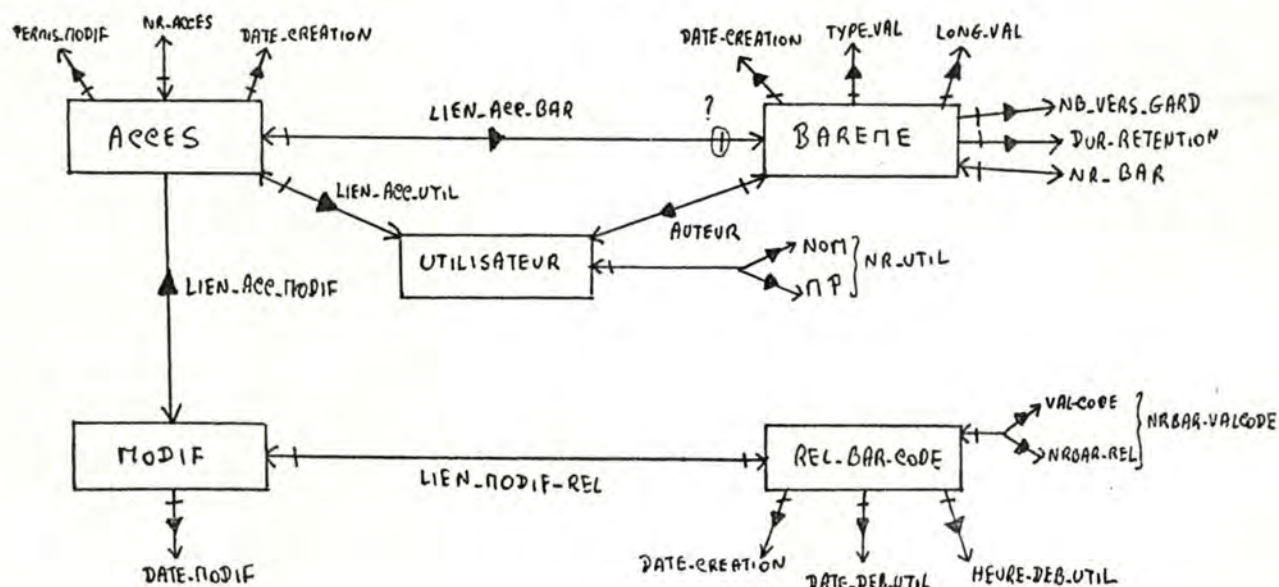


Par T1 et T2, au type d'entité ACCES (et attributs) correspond le type d'article ACCES (et ses items); et au type d'entité UTILISATEUR (et ses attributs) correspond le type d'article UTILISATEUR et ses trois items NOM, MP, et CHEF. Les deux premiers forment une clé d'accès identifiante de ce type d'article que l'on baptisera NR-UTIL.

Selon T3, au type d'association LIEN-ACC-UTIL correspond le type de chemin LIEN-ACC-UTIL entre les types d'articles ACCES et UTILISATEUR; au type d'association AUTEUR correspond le type de chemin AUTEUR entre les types d'articles BAREME et UTILISATEUR; et au type d'association LIEN-BAR-ACC correspond le type de chemin LIEN-BAR-ACC entre les types d'articles BAREME et ACCES.

Selon T4, au type d'association MODIF (avec attribut) correspond le type d'article MODIF, son item et deux types de chemins LIEN-ACC-MODIF entre les types d'articles ACCES et MODIF, et LIEN-MODIF-REL entre les types d'articles MODIF et REL-BAR-CODE.

Voici donc le schéma obtenu pour la gestion de la sécurité :



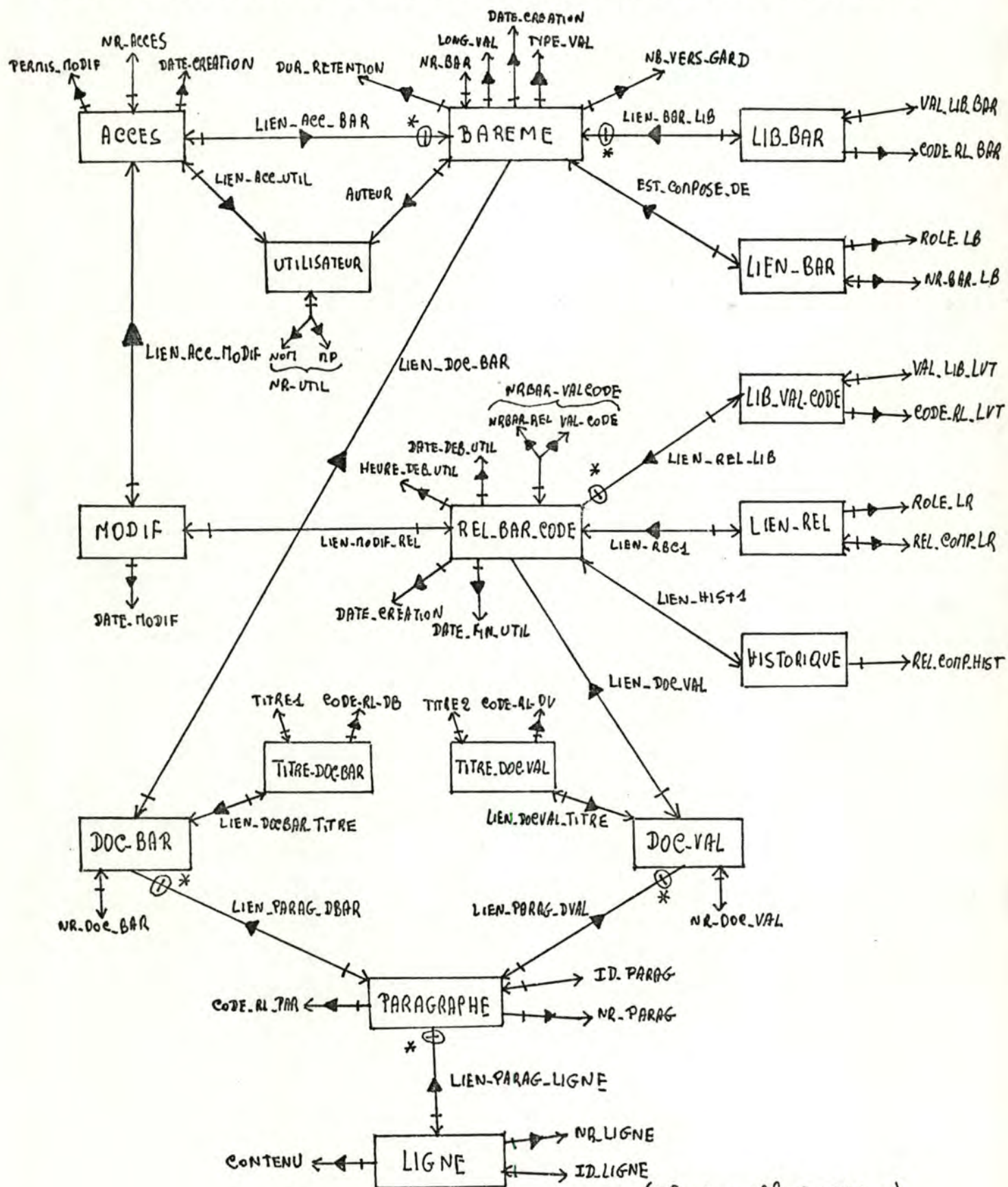
Selon T1 et T2, au type d'entité DOC-BAR (et ses attributs) correspond le type d'article DOC-BAR (et ses items); au type d'entité DOC-VAL (et ses attributs) correspond le type d'article DOC-VAL (et son item); au type d'entité PARAGRAPHE (et ses attributs) correspond le type d'article PARAGRAPHE (et ses items); au type d'entité LIGNE (et attributs) correspond le type d'article LIGNE (et items); au type d'entité TITRE-DOC-BAR (et ses attributs) correspond le type d'article TITRE-DOC-BAR (et items); et au type d'entité TITRE-DOC-VAL (et attributs) correspond le type d'article TITRE-DOC-VAL (et items).

Selon T3, au type d'association LIEN-DOC-BAR correspond le type de chemin LIEN-DOC-BAR entre les types d'articles BAREME et DOC-BAR; au type d'association LIEN-DOC-VAL correspond le type de chemin LIEN-DOC-VAL entre les types d'articles REL-BAR-CODE et DOC-VAL; au type d'association LIEN-PARAG-LIGNE correspond le type de chemin LIEN-PARAG-LIGNE entre les types d'articles LIGNE et PARAGRAPHE; au type d'association LIEN-PARAG-DBAR correspond le type de chemin LIEN-PARAG-DBAR entre les types d'articles DOC-BAR et PARAGRAPHE; au type d'association LIEN-PARAG-DVAL correspond le type de chemin LIEN-PARAG-DVAL entre les types d'articles DOC-VAL et PARAGRAPHE; au type d'association LIEN-DOCBAR-TITRE correspond le type de chemin LIEN-DOCBAR-TITRE entre les types d'articles DOC-BAR et TITRE-DOC-BAR; et au type d'association LIEN-DOCVAL-TITRE correspond le type de chemin LIEN-DOCVAL-TITRE entre les types d'articles DOC-VAL et TITRE-DOC-VAL.

On peut maintenant reconstituer le schéma d'accès complet en ce compris les types d'articles et de chemins que l'on vient d'exposer et qui concernent la documentation.

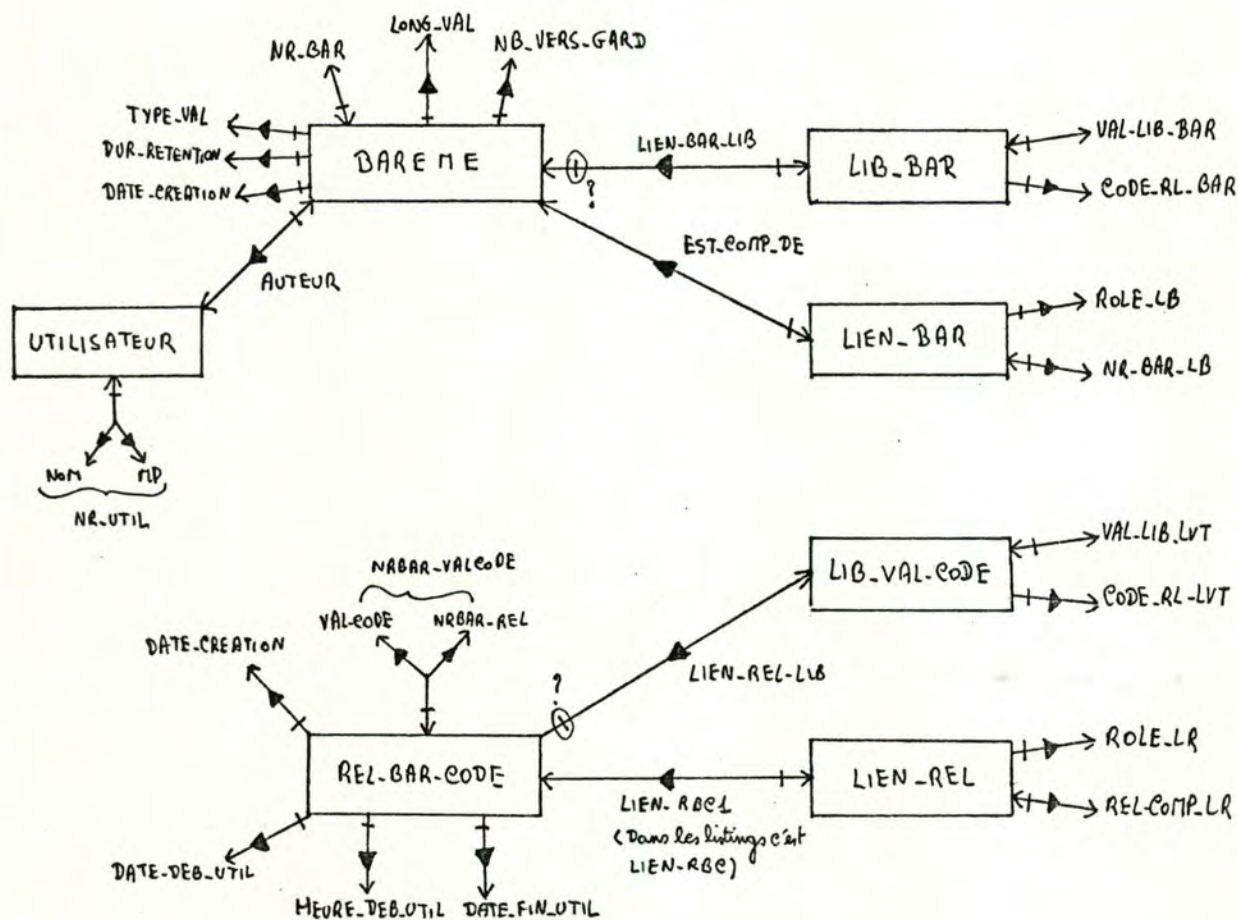
b) Le schéma.

Le schéma d'accès compatible CODASYL dérive du schéma E/A se représente donc ainsi :



(* Pas compatible CODASYL, à gérer par les programmes.)

Ce schéma n'a pas été complètement implémenté (cfr page 52). Le sous-schéma implémenté se présente, pour sa part, ainsi :



(? = pas compatible CODASYL, à gérer par programmation)

Rappel : $NR-UTIL = (NOM, MP)$; $NRBAR-VALCODE = (NRBAR-REL, VAL-CODE)$;
 $NR-BAR-LB (:LIEN-BAR) = NR-BAR (:BAREME)$;
 $REL-COMP-LR (:LIEN-REL) = NRBAR-VALCODE (:REL-BAR-CODE)$;
 $NRBAR-REL (:REL-BAR-CODE) = NR-BAR (:BAREME)$.

Remarque : Comme on l'a réalisé ici, pour la sécurité, un utilisateur n'a accès qu'aux barèmes qu'il a créés et uniquement ceux-là un peu comme un système de "directories". Précisons aussi que la concurrence d'accès est gérée dans les programmes.

3.2 Architecture du logiciel.

- a) Le schéma modulaire et les règles qui président
 //////////////////////////////////////
 à sa constitution.
 //////////////////////////////////////

Tout d'abord, il est bon de se rappeler les règles qui régissent l'architecture d'un bon logiciel.

On dit qu'une structure est modulaire si :

- 1) les attributs (spécifications, mode de réalisation) de chaque composant (module) peuvent être définis simplement et précisément;
- 2) et si chaque module offre :
 - une forte capacité de cacher de l'information (cela permet et pouvoir procéder par niveaux d'abstraction);
 - une forte cohésion interne;
 - un faible degré de couplage.

Ces trois derniers points sont aussi les critères d'évaluation d'une bonne modularisation.

Les qualités d'un bon logiciel sont sa fiabilité, sa maintenabilité, la qualité de la documentation, la réutilisabilité, la portabilité, la performance, la convivialité avec l'utilisateur, la sécurité et la protection de l'information.

Ces règles sont issues du cours D'A.VAN-LANSWEERDE intitulé "Méthodologie de développement de logiciels", donné en deuxième licence et maîtrise en informatique (FUNDP). Procédons maintenant à la découpe modulaire du système.

Notre schéma se constitue de deux niveaux. Le niveau du bas comprend quatre ensembles bien distincts selon leur fonction. Chacun de ces ensembles constitue donc un module particulier. Ces quatre-ci sont ce qu'on appelle des modules de "données" car ils s'occupent de la gestion des données de notre schéma. Le premier est le module "SECURITE". Il a pour but de gérer la notion de sécurité d'accès dans le système. On y trouvera les notions d'utilisateur, d'accès à un barème, de droit d'accès, de droit de modification, de contrôle d'accès, etc. En bref s'y trouve tout ce qui touche à la sécurité.

Toute modification possible qui concerne ce sujet se localisera donc ici. On voit bien, par là, l'avantage de regrouper dans un seul ensemble toutes les primitives qui touchent à la protection du système. Toutes ces primitives travaillent sur une partie du schéma. Cela est aussi avantageux par le fait que ce module renferme le secret de la constitution de cette partie du schéma. Il est le seul à connaître la structure sur laquelle il travaille et il la cache donc vis-à-vis de l'extérieur. De plus, il est extensible et modifiable à souhait sans devoir aller voir partout ailleurs dans le système.

Le deuxième est le module "BAREME". Il a pour but de rassembler toutes les primitives qui gèrent les barèmes et travaillent sur ceux-ci. Ces primitives auront pour but de créer, modifier, supprimer des barèmes etc. A nouveau toute modification qui concerne les barèmes sera localisée dans ce module. Il est donc aussi extensible et modifiable à souhait sans conséquence pour le reste du système.

Le troisième est le module "DOCUMENTATION". Le souci sous-jacent est ici de rassembler toutes les tâches qui touchent à la documentation du système. Par ce fait, on a l'avantage de constituer une base de documentation dont toute la gestion est l'objet de ce module. Celui-ci renferme bien-sûr le secret de la constitution de cette base de documentation. Cet avantage permet comme pour les deux modules précédents de pouvoir procéder par niveau d'abstraction et de faciliter la modification ou l'extension du module en question.

Le quatrième est le module "ES" ou "Entrées-Sorties". Il est en effet toujours bon de rassembler dans un seul module tous les problèmes d'entrées/sorties interactives. Contrairement aux trois autres, ce module ne travaille pas sur le schéma. Ici, il s'agit de l'écran. L'avantage majeur de ce rassemblement est de pouvoir cacher le choix de réalisation de ces entrées/sorties. Souvent, sur les systèmes d'ordinateur des entreprises, le programmeur peut utiliser à cette fin des questionnaires d'écran qui lui facilitent grandement la tâche. Ici les modules qui appellent ce module ne doivent rien savoir quand au choix de réalisation qui en a été fait. Pour utiliser un questionnaire d'écran, il faut parfaitement l'apprendre et le maîtriser. Par manque de temps, on choisira de réaliser une gestion d'écran assez simple tout en étant bien conscient que notre architecture logicielle pourrait nous permettre l'utilisation d'un tel questionnaire. On peut encore ajouter que ce module est extensible et modifiable à volonté et qu'il décharge tous les autres modules des problèmes d'entrées/sorties.

En fait, on voit ici que, dans cette découpe modulaire, on satisfait aux trois critères d'évaluation d'une bonne modularisation :

- * chaque module a une tâche bien précise à réaliser et possède une forte cohésion interne;
- * leurs interactions sont faibles et ils possèdent chacun une forte capacité de cacher de l'information;
- * il en résultera lors du développement un faible degré de couplage. Les arguments sont nombreux mais pas compliqués.

Ces quatre modules sont la base de notre architecture. Ils gèrent donc les données du système en renfermant leur secret respectif. Il constitue en quelque sorte un certain nombre d'outils que les modules en amont pourront utiliser à souhait .

Maintenant que nos données sont gérées, il nous reste à nous occuper de nos traitements c-à-d de ce qu'on appellera les modules "du haut" ou modules des "traitements".

Comme on l'a déjà vu, deux sortes de traitements sont à réaliser : les traitements interactifs et ceux par programmes. Il en résulte deux modules distincts !

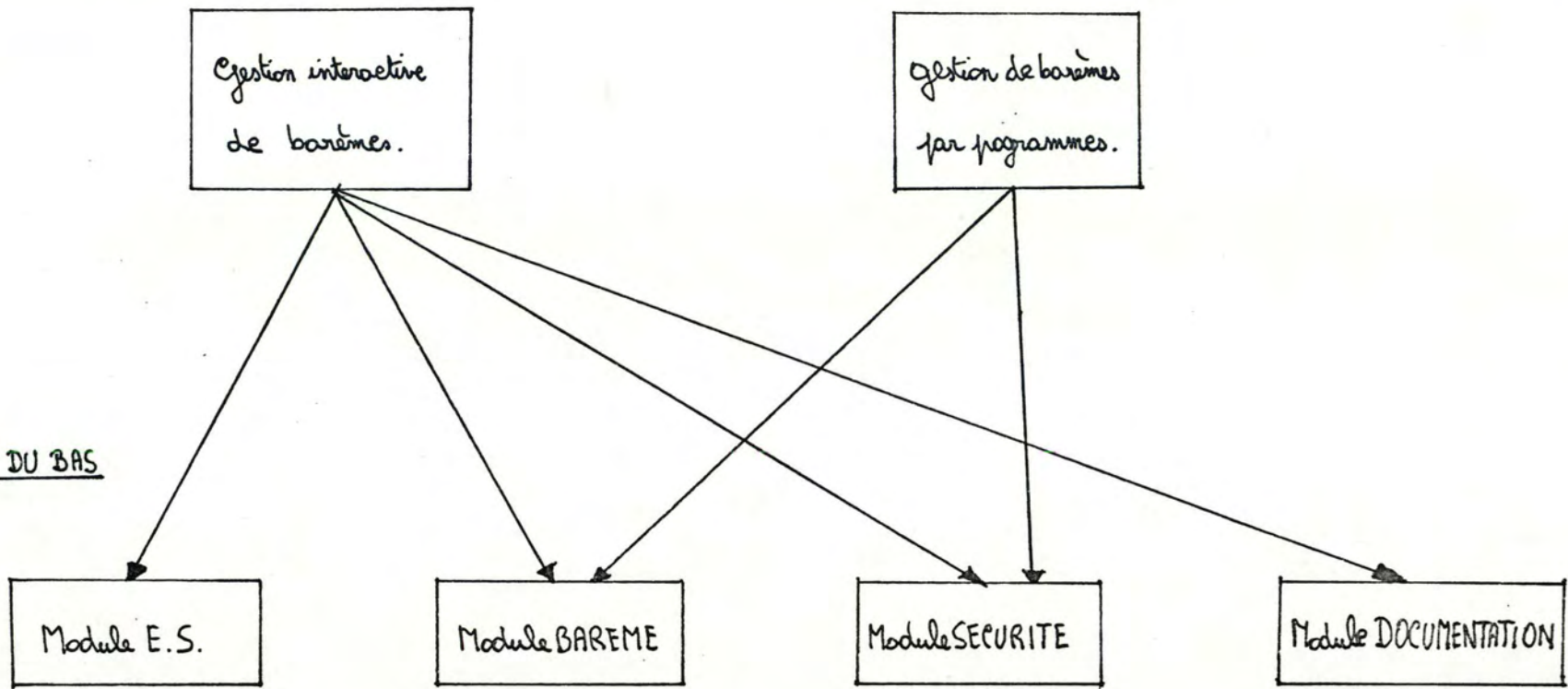
Le premier ou "module de gestion interactive de barèmes" a pour but de réaliser, en utilisant les modules du bas, les traitements qui ont été déterminés dans l'analyse fonctionnelle pour la partie interactive. Un appel à ce module entraîne une identification interactive de l'utilisateur qui veut entrer dans le système et en cas de succès de celle-ci, l'affichage d'un menu général dans lequel on peut lire les différentes options ou traitements disponibles. Tous ceux-ci sont alors exécutables à souhait et tout cela en interaction avec l'utilisateur.

Le second module de traitements est le "module de gestion de barèmes par programmes". Dans l'analyse fonctionnelle, on a identifié deux traitements qui peuvent être appelés par des programmes divers. Il s'agit de deux traitements de consultation que ce module s'occupe de réaliser en utilisant les modules de données à l'exception du module "ES".

Voici maintenant à la page suivante le schéma de notre architecture modulaire avec les différentes relations "utilise" appropriées. Ensuite suivront les spécifications des différents modules en commençant par le niveau du bas.

NIVEAU DU HAUT

NIVEAU DU BAS



(Les flèches signifient : "UTILISE")

Le schéma modulaire.

b) Description des modules.

////////////////////////////////////

Le module BAREME.

Ci-dessous, sont présentées les quatorze primitives de ce module. Elles ont pour but de gérer les barèmes dans le système. Ce sont donc des primitives qui gèrent les données qui concernent les barèmes. On précisera pour chacune d'elles, la fonction à réaliser (déterminée dans l'analyse fonct.) auquel elle sera utile.

Mais au préalable, nous prendrons connaissance de l'interface du module par lequel celui-ci transmettra ses informations.

Par facilité, pour la description de cet interface, on ne procédera pas à une séparation de ses arguments en "entrées-sorties". Pour en avoir le détail, on consultera directement la description des primitives du module qui suit celle de l'interface.

1/ Interface et objectif du module.

L'interface du module est constitué de :

& NUM-PROC est le numéro de la primitive à exécuter.

& ID-BAR est l'identificateur d'un barème.

& EXISTE est un indicateur de travail dont la valeur doit être 1 si une certaine chose existe dans le système et 0 sinon.

& LIBS-BAR est un tableau destiné à contenir les libellés d'un barème et leurs régimes linguistiques associés.

& NL est la longueur de ce tableau.

& DT-CR est la date de création d'un barème ou d'une valeur de barème.

& TP-VAL est le type de valeur de code associé à un barème.

& LG-VAL est la longueur maximale d'une valeur de code.

& NB-VERS est le nombre de versions d'une valeur que l'on peut garder en historique.

& DR-RET est la durée de rétention des valeurs d'un barème.

& TAB-BARS est un tableau d'identificateurs de barèmes composants auxquels on a associé leurs rôles dans la composition et leurs durées de rétention.

& NB est la longueur de ce tableau.

& TAB-IDS est un tableau destiné à contenir un ensemble d'identificateurs de barèmes.

& NC est la longueur de ce tableau.

& ID-REL est l'identificateur d'une valeur de barème.

Il est composé de l'identificateur du barème auquel cette valeur appartient et de sa valeur de code.

& TAB-RELS est un tableau d'identificateur de valeurs de barème composantes auxquelles sont associées leurs rôles dans la composition.

& NV est la longueur de ce tableau.

& LIBLU est le libellé d'un barème objet d'un traitement.

& DT-DB est la date de début d'utilisation d'une valeur de barème.

& HR-DB est son heure de début d'utilisation.

& DT-FN est sa date de fin d'utilisation.

& TABLEAU-VALS est un tableau de valeurs de code résultant d'une recherche selon critère ou d'un affichage de barème.

& NLIG est la longueur de ce tableau.

& TERMI est un indicateur qui stipule si une recherche selon critère est terminée ou pas.

& W est le numéro du tableau de valeurs TABLEAU-VALS que l'on fournit.

& CLE est un identificateur de valeur de barème que l'on doit retenir afin de pouvoir poursuivre une recherche en cours.

Objectif :

Ce module a pour but d'identifier et d'exécuter la primitive dont le numéro est spécifié dans l'argument NUM-PROC. Précisons qu'à chaque primitive, nous donnons un numéro qui l'identifie. Spécifions maintenant chacune de ces primitives.

2/ Spécification des primitives.

Primitive 1 : CR-TETE-BAR.

Entrées : ID-BAR, DT-CR, TP-VAL, LG-VAL, NB-VERS, DR-RET, LIBS-BAR, NL, TAB-BARS, NB .

Précond : ID-BAR inexistant; DT-CR = date du système; TP-VAL vaut "AN" ou "AB" ou "NU"; LG-VAL compris entre 01 et 06; NB-VERS compris entre 00 et 20; DR-RET est compris entre 0001 et 9999; NL entre 01 et 10; NB entre 00 et 10; LIBS-BAR contient NL entrées non vides et TAB-BARS contient NB entrées non vides.

Sorties : /

Postcond : un nouveau barème existe dans le système.

Objectif : Il s'agit de créer un nouvel en-tête de barème dans le système. Cet en-tête se matérialise par toutes les valeurs des arguments que l'on trouve en entrée. Cette primitive sera utile à la fonction "création de barème" (cfr 2.4.a.1).

Primitive 2 : OBTENIR-BAR.

Entrées : ID-BAR .

Précond : ID-BAR existant.

Sorties : DATE-CREATION, TYPE-VAL , LONG-VAL, NBR-VERS-GARD ,
DUR-RETENTION, LIBS-BAR, NL, TAB-BARS, NB.

Postcond : les arguments de sortie doivent contenir des valeurs qui correspondent au ID-BAR d'entrée.

Objectif : Le problème est ici de transférer dans les arguments de sortie, les valeurs de l'en-tête du barème dont le numéro est ID-BAR. Cette primitive sera utile à la fonction "Affichage de barème". (cfr 2.4.a.4)

Primitive 3 : EXIST-BAR.

Entrée : ID-BAR.

Sortie : EXISTE.

Postcond : EXIST = 1 ou 0.

Objectif : Il s'agit de contrôler s'il existe un barème dont l'identificateur est ID-BAR dans le système. Si c'est le cas, EXISTE doit être mis à 1 sinon à 0. Cette primitive sera utile à la fonction "Création de barème". (cfr 2.4.a.1)

Primitive 4 : OBTENIR-ALL-IDBAR.

Entrées : /

Sorties : TAB-IDS, NC.

Postcond : NC compris entre 01 et 99; TAB-IDS contient NC entrées non vides c-à-d NC identificateurs de barèmes distincts et existants.

Objectif : le but est ici de mettre dans le tableau TAB-IDS , tous les identificateurs de tous les barèmes du système. Cette primitive sera utile à la fonction "Obtention de tous les noms de barèmes". (cfr 2.4.a.6)

Primitive 5 : EXIST-LIB-BAR.

Entrée : LIBLU.

Sorties : ID-BAR, EXISTE.

Postcond : EXIST = 1 ou 0; et si c'est 1 alors, ID-BAR contient l'identificateur de barème associé à LIBLU.

Objectif : Le but est ici de contrôler dans le système si le libellé du barème donné en entrée dans LIBLU existe dans le système. Si c'est le cas, alors EXISTE sera mis à 1 et on devra fournir dans ID-BAR l'identificateur du barème qui correspond à ce libellé, sinon EXISTE sera mis à 0. Cette primitive sera utile à plusieurs fonctions. (cfr 2.4.a.2,3,4,5,7)

Primitive 6 : OBTENIR-COMPOSE.

Entrée : ID-BAR .

Précond : ID-BAR existant.

Sorties : TAB-IDS, NC.

Postcond : NC compris entre 01 et 99; et TAB-IDS contient NC entrées non vides et qui répondent à l'objectif.

Objectif : Le but est d'obtenir dans TAB-IDS les identificateurs de tous les barèmes qui admettent le barème dont l'identificateur est ID-BAR, comme composant. Cette primitive sera utile à la fonction " Obtenir la composition d'un barème". (cfr 2.4.a.5).

Primitive 7 : CR-VAL-BAR.

Entrées : ID-BAR, ID-REL, DT-CR, DT-DB, HR-DB, LIBS-BAR, NL, TAB-RELS, NV.

Précond : ID-BAR existant; ID-REL inexistant; DT-CR = date du système; DT-CR, DT-DB, DT-FN sont des dates du calendrier, HR-DB est une heure valide; $DT-CR \leq DT-DB \leq DT-FN$; NL compris entre 01 et 10; NV compris entre 00 et 10; LIBS-BAR contient NL entrées non vides; TAB-RELS contient NV valeurs composantes correctes et existantes.

Sorties : /

Postcond : une nouvelle valeur de barème est créée dans le barème d'identificateur ID-BAR.

Objectif : Il s'agit de créer une nouvelle valeur, dont la composition est donnée en entrée, dans le barème dont l'identificateur est ID-BAR. Cette primitive est utile à la fonction "Création d'une valeur de barème. (cfr 2.4.a.2)

Primitive 8 : OBTENIR-VAL-BAR.

Entrée : ID-REL.

Sorties : EXISTE, DT-CR, DT-DB, HR-DB, DT-FN, LIBS-BAR, NL, TAB-RELS, NV.

Postcond : EXISTE = 0 ou 1; et si c'est 1 alors les autres arguments de sortie doivent contenir les renseignements correspondant à ID-REL donné en entrée.

Objectif : Si la valeur de barème dont la clé est ID-REL existe dans le système, alors EXISTE doit être mis

à 1 et la composition de cette valeur doit être copiée dans les arguments de sortie décrits ci-dessus sinon EXISTE doit être mis à 0. Cette primitive sera utile à la fonction "Obtenir les libellés d'une valeur de code". (cfr 2.4.a.3)

Primitive 9 : PAS-DE-BAREMES.

Entrée : /

Sortie : EXISTE.

Postcond : EXISTE = 0 ou 1.

Objectif : Il faut ici contrôler s'il existe au moins un barème dans le système. Si c'est le cas, EXISTE doit valoir 1 sinon 0. Cette primitive sera utile à plusieurs fonctions. (cfr 2.4.a.2 -> 9)

Primitive 10 : PAS-DE-VALEURS.

Entrée : ID-BAR .

Précond : ID-BAR existant.

Sortie : EXISTE.

Postcond : EXISTE = 0 ou 1.

Objectif : S'il existe au moins une valeur dans le barème dont l'identificateur est ID-BAR, alors EXISTE doit valoir 1 sinon 0. Cette primitive sera utile à plusieurs fonctions. (cfr 2.4.a.2,3,4,7)

Primitive 11 : GIVE-HT-VALS.

Entrées : ID-BAR , CLE et W .

Précond : ID-BAR existant; $W \geq 1$; et si $W = 1$ alors CLE est vide sinon CLE contient un identificateur de valeur du barème de n° ID-BAR.

Sorties : TABLEAU-VALS, NLIG, CLE, TERMI.

Postcond : NLIG compris entre 00 et 08; TABLEAU-VALS contient NLIG entrées dont les valeurs sont conformes à l'objectif, tout comme CLE et TERMI.

Objectif : Il s'agit de remplir TABLEAU-VALS avec huit valeurs d'actualité du barème d'identificateur ID-BAR. Si W est égal à 1, alors on prendra les valeurs à partir de la première. Si W est plus grand que 1, alors on prendra les valeurs qui suivent la valeur dont l'identificateur est CLE. On terminera en mémorisant dans CLE l'identificateur de la valeur qui concerne la dernière composante non vide de TABLEAU-VALS. La longueur de ce tableau est NLIG. Et si celle-ci est plus petite que 8, alors c'est qu'on est arrivé à la dernière valeur et TERMI doit valoir 1. Cette primitive sera utile à la fonction "Affichage d'un barème". (cfr 2.4.a.4)

Primitive n° 12 : RECH-HT-VALS.

Entrées : ID-BAR, TAB-RELS, NV, CLE, W.

Précond : ID-BAR existant; TAB-RELS contient des valeurs existantes et d'actualité des barèmes composants du barème n° ID-BAR; $01 \leq NV \leq 10$; pour CLE et W même chose que la primitive 11.

Sorties : TABLEAU-VALS, NLIQ, W, CLE, TERMI.

Postcond : NLIQ compris entre 00 et 08; TABLEAU-VALS contient NLIQ entrées dont les valeurs sont conformes à l'objectif, tout comme CLE et TERMI.

Objectif : Il est identique à la primitive précédente excepté qu'ici, on ne désire avoir que les valeurs de barème d'actualité, qui contiennent toutes les valeurs de code spécifiées dans TAB-RELS. Le résultat est mis dans TABLEAU-VALS. Remarquons que dans ce tableau, on y copie que les valeurs de code qui constituent les lignes de barème qui résultent de la recherche. Cette primitive sera utile à la fonction "Recherche selon critère". (cfr 2.4.a.7)

Primitive 13 : SUPPRESSION-VALEURS.

Entrée : ID-BAR .

Précond : ID-BAR existant.

Postcond : le barème de n° ID-BAR ne contient plus de valeurs périmées.

Objectif : Il s'agit de supprimer, dans le barème d'identificateur ID-BAR, toutes les valeurs qui sont périmées. Cela sera utile à "Nettoyage système".(cfr 2.4.a.9)

Primitive 14 : SUPPRESSION-BAREMES.

Entrée : ID-BAR .

Précond : ID-BAR existant.

Postcond : le système ne contient plus le barème de n° ID-BAR.

Objectif : Le but est ici de supprimer du système l'en-tête du barème dont l'identificateur est ID-BAR. Cela sera utile à "Nettoyage système".(cfr 2.4.a.9)

Le module Entrées - Sorties.

Ce module est de loin le moins important du système. Il ne présente que peu d'intérêt à côté des autres. Il comprend des primitives de lecture et d'écriture d'informations à l'écran. Bien entendu, les lectures sont accompagnées des validations syntaxiques appropriées.

Dans ce module, ont été créées vingt-sept petites primitives. Afin d'en éviter une description inutilement longue, après avoir vu l'interface de ce module, nous décrirons ensuite quelques-unes de ces primitives les plus importantes, et dès lors, nous invitons le lecteur à consulter l'annexe 2 s'il désire parcourir l'ensemble de celles-ci.

Tout comme le module précédent, chaque primitive porte un numéro qui l'identifie et nous procéderons de la même manière pour la description de l'interface. Précisons enfin que c'est ici que se trouve la liste des messages du système et qu'à chacun de ceux-ci, on a attribué un numéro identifiant.

1/ Interface et objectif du module.

L'interface du module est constitué de :

- & NUM-PROC est le numéro de la primitive à exécuter.
- & NUM-MESS est le numéro du message à sortir à l'écran.
- & ID-BAR est un identificateur de barème.
- & LIBS-BAR est un tableau destiné à contenir les libellés d'un barème et leurs régimes linguistiques associés.
- & NL est la longueur de ce tableau.
- & TAB-BARS est un tableau d'identificateurs de barèmes composants auxquels on a associé leurs rôles dans la composition et leur durée de rétention.
- & NB est la longueur de ce tableau.

- & NBP est une variable auxiliaire du type de NB.
- & DT-CR est une date de création d'un barème ou d'une valeur.
- & TP-VAL est le type des valeurs de code d'un barème.
- & LG-VAL est la longueur maximale des valeurs de code d'un barème.
- & NB-VERS est le nombre maximum de versions d'une valeur que l'on peut garder en historique.
- & DR-RET est la durée de rétention des valeurs d'un barème.
- & LIBLU est un libellé de barème lu à l'écran.
- & EXISTE est un indicateur de travail dont la valeur doit être 1 si une certaine chose existe dans le système et sinon 0.
- & FINI est un indicateur dont la valeur est 1 si c'est la fin de quelque chose et sinon 0.
- & I est un indice de boucle.
- & OPT-MENU est un numéro identifiant une option dans un menu.
- & DT-DB est la date de début d'utilisation d'une valeur de barème.
- & HR-DB est son heure de début d'utilisation.
- & DT-FN est sa date de fin d'utilisation.
- & VAL-CODE est une valeur de code.
- & ID-REL est un identificateur de valeur de barème composé de sa valeur de code et de l'identificateur de ce barème.
- & TAB-RELS est un tableau d'identificateur de valeurs de barème composantes auxquelles sont associées leurs rôles dans la composition.
- & NV est la longueur de ce tableau.
- & REP est une réponse dont la valeur est "OUI" ou "NON".
- & TABLEAU-VALS est un tableau de valeurs de code résultant d'une recherche selon critère ou d'un affichage de barème.
- & NLI est la longueur de ce tableau.

& TABLEAU-LIBS est un tableau de libellés de barème.

& LIBBAR est un libellé de barème.

& ID-UTIL est l'identificateur d'un utilisateur constitué de son nom et de son mot de passe.

Objectif :

Ce module a pour but d'identifier et d'exécuter la primitive dont le numéro est spécifié dans NUM-PROC.

Voyons maintenant quelques exemples de ces primitives !

2/ Exemples de primitives.

Primitive 0 : ALL-MESSAGE. (Le numéro de cette primitive est ZERO)

Entrée : NUM-MESS .

Précond : NUM-MESS existant c-à-d entre 01 et 86.

Postcond : le message n° NUM-MESS est affiché.

Sortie : un message à l'écran.

Objectif : le but est de sortir à l'écran le message dont le numéro est NUM-MESS. Exemples de messages :

- "Barème inexistant";
- "Recherche impossible sur les barèmes simples!";
- Etc.

Primitive 5 : LECT-BAR-RENS.

Entrées : /

Sorties : DT-CR, TP-VAL, LG-VAL, NB-VERS, NL.

Postcond : DT-CR = date du système; TP-VAL = "AN" ou "NU" ou "AB"; LG-VAL compris entre 01 et 06; NB-VERS compris entre 00 et 20; NL compris entre 01 et 10.

Objectif : Lire et valider les arguments décrits en sortie.

Primitive 14 : SORTIR-BAR.

Entrées : ID-UTIL, LIBS-BAR, ID-BAR, DT-CR, TP-VAL, LG-VAL, NB-VERS, DR-RET.

Précond : ID-UTIL, LIBS-BAR, ID-BAR existants; et tous les autres arguments doivent être corrects syntaxiquement et sémantiquement.

Sorties : /

Postcond : les entrées sont affichées à l'écran.

Objectif : Le but est de sortir un en-tête de barème à l'écran c-à-d toutes les valeurs des arguments donnés en entrée.

Primitive 17 : LECT-REL-RENS-LIBS.

Entrées : /

Sorties : NL, TAB-LIBS.

Postcond : NL est compris entre 01 et 10; LIBS-BAR contient NL entrées non vides.

Objectif : Il s'agit de lire et de valider NL, et ensuite de lire NL libellés.

Primitive 26 : IDENTIFIER.

Entrées : /

Sortie : ID-UTIL.

Postcond : ID-UTIL non vide.

Objectif : Il s'agit de procéder à l'écran à l'identification d'un utilisateur. Elle se compose d'un nom et d'un mot de passe. La lecture de ce mot de passe doit se faire sans écho au terminal.

Le module SECURITE.

Comme cela a déjà été précisé, par manque de temps, la sécurité prévue n'a pu subir qu'un début de développement. Nous verrons donc d'abord les primitives qui ont été réalisées et nous en proposerons ensuite quelques autres qui devraient être utiles à notre système. Précisons que ce module a un caractère exclusivement technique.

Mais d'abord, penchons - nous sur l'interface de ce module.

1/ Interface et objectif du module.

L'interface du module est constitué de :

& NUM-PROC est le numéro de la primitive qu'il faut exécuter.

& ID-UTIL est l'identificateur d'un utilisateur. Il est composé d'un nom et d'un mot de passe.

& ID-BAR est un identificateur de barème sur lequel doit porter le traitement.

& TAB est un tableau destiné à contenir des identificateurs de barèmes.

& N est la longueur de ce tableau.

& EXISTE est un indicateur d'existence. Il vaut 1 en cas d'existence et sinon 0.

Objectif :

Le but de ce module est d'exécuter la primitive dont le numéro est NUM-PROC et donc de répondre à la demande du module appelant.

2/ Spécification des primitives implémentées.

Primitive 1 : EXIT-UTILISATEUR.

Entrée : ID-UTIL.

Sortie : EXISTE.

Postcond : EXISTE = 1 ou 0 conformément à l'objectif.

Objectif : Le but est de vérifier dans le système si l'utilisateur dont l'identificateur est ID-UTIL existe ou non. Si c'est le cas alors EXISTE devra valoir 1 sinon 0.

Primitive 2 : EXIST-ACCES.

Entrées : ID-BAR, ID-UTIL.

Précond : ID-UTIL, ID-BAR (supposés corrects, existants).

Sortie : EXISTE.

Postcond : EXISTE = 1 ou 0 conformément à l'objectif.

Objectif : Il faut vérifier dans le système si l'utilisateur dont l'identificateur est ID-UTIL a accès au barème identifié par ID-BAR. Si c'est le cas, alors EXISTE doit valoir 1 sinon 0.

Primitive 3 : CREER-ACCES.

Entrées : ID-BAR, ID-UTIL.

Précond : ID-UTIL, ID-BAR (supposés corrects, existants).

Postcond : l'accès décrit dans l'objectif est créé.

Objectif : Il s'agit de créer un accès au barème identifié par ID-BAR pour l'utilisateur dont l'identificateur est ID-UTIL.

Primitive 4 : GIVE-ACCES.

Entrée : ID-UTIL.

Précond : ID-UTIL (supposé existant).

Sorties : TAB, N.

Postcond : N compris entre 00 et 99; TAB contient N entrées non vides et conformes à l'objectif.

Objectif : Le but est de mettre dans TAB tous les identificateurs des barèmes auxquels l'utilisateur identifié par ID-UTIL a accès.

3/ Spécification de primitives non implémentées.

Primitive 5 : CREER-UTILISATEUR.

Entrée : ID-UTIL.

Précond : ID-UTIL (supposé inexistant).

Postcond : un nouvel utilisateur d'identificateur ID-UTIL existe dans la base de données.

Objectif : Il s'agit de créer un nouvel utilisateur dans le système.

Primitive 6 : SUPPRIMER-UTILISATEUR.

Entrée : ID-UTIL.

Précond : ID-UTIL (supposé existant).

Postcond : l'utilisateur d'identificateur ID-UTIL n'existe plus dans la base de données.

Objectif : Ici, il faut supprimer du système l'utilisateur dont l'identificateur est ID-UTIL.

Primitive 7 : SUPPRIMER-ACCES.

Entrées : ID-UTIL, ID-BAR.

Précond : ID-UTIL, ID-BAR (supposés existants).

Postcond : l'accès qui porte sur ID-UTIL et ID-BAR n'existe plus dans la base de données.

Objectif : Il s'agit de supprimer l'accès au barème de numéro identifiant ID-BAR à l'utilisateur identifié par ID-UTIL.

Primitive 8 : EST-CHEF.

Entrée : ID-UTIL.

Précond : ID-UTIL (supposé existant).

Sortie : REP.

Postcond : REP = "OUI" ou "NON" conformément à l'objectif.

Objectif : Le but est de vérifier si l'utilisateur identifié par ID-UTIL est le chef du système ou pas. Si c'est le cas alors REP devra contenir "OUI" sinon "NON".

Le module DOCUMENTATION.

La gestion de la documentation étant une tâche secondaire, il a été décidé de ne pas l'implémenter dans notre prototype faute de temps.

Ce module n'a donc pas été réalisé. Néanmoins, suit ci-dessous une proposition d'interface et de primitives qui seraient utiles à notre système. Tout comme les autres modules, on se propose de numéroter ces primitives et de décrire l'interface de la même façon.

1/ Interface et objectif du module.

L'interface du module est constitué de :

\$ ID-DOC est un identificateur de documentation.

& ID-BAR est un identificateur de barème.

& ID-REL est un identificateur de valeurs de barème.

& TITRE est un titre de documentation.

& EXISTE est un indicateur d'existence. (Cfr modules précédents)

& TITRES-DOC est un tableau destiné à contenir les titres d' une documentation. (NT est la longueur de ce tableau)

& TAB-DOC est un tableau destiné à contenir une documentation.

(LD est sa longueur).

Objectif :

Ce module a pour but d'identifier et d'exécuter la primitive dont le numéro est spécifié dans NUM-PROC.

Voyons maintenant la spécification de nos primitives.

2/ Spécification des primitives.

Primitive 1 : CREER-DOC-BAR.

Entrées : ID-BAR, ID-DOC , TAB-DOC,LD,TITRES-DOC,NT.

Précond : ID-BAR existant, ID-DOC inexistant;NT compris entre 01 et 10; LD compris entre 01 et 120 ; TITRES-DOC contient NT entrées inexistantes et TAB-DOC en contient LD quelconques et non vides.

Postcond : la documentation donnée en entrée est créée dans la base de données.

Objectif : Il s'agit de créer une nouvelle documentation qui concerne le barème identifié par ID-BAR dans le système.

Primitive 2 : CREER-DOC-VAL.

Entrées : ID-REL, ID-DOC , TAB-DOC,LD,TITRES-DOC,NT.

Précond : ID-REL existant, ID-DOC inexistant;NT compris entre 01 et 10; LD compris entre 01 et 120 ; TITRES-DOC contient NT entrées inexistantes et TAB-DOC en contient LD quelconques et non vides.

Postcond : la documentation donnée en entrée est créée dans la base de données.

Objectif : Il s'agit de créer une nouvelle documentation qui concerne la valeur de barème identifiée par ID-REL dans le système.

Primitive 3 : EXIST-DOC-BAR.

Entrée : TITRE.

Sorties : ID-DOC, EXISTE.

Postcond : EXISTE = 1 ou 0 conformément à l'objectif. Et si c'est 1, alors ID-DOC contient l'identificateur de documentation correspondant à TITRE.

Objectif : Il s'agit de vérifier s'il existe une documentation (portant sur un barème) dont le titre TITRE est donné en entrée. Si c'est le cas, alors EXISTE doit valoir 1 et ID-DOC doit comporter l'identificateur de cette documentation, sinon EXISTE doit valoir 0.

Primitive 4 : EXIST-DOC-VAL.

Entrée : TITRE.

Sorties : ID-DOC, EXISTE.

Postcond : EXISTE = 1 ou 0 conformément à l'objectif. Et si c'est 1, alors ID-DOC contient l'identificateur de documentation correspondant à TITRE.

Objectif : Idem que la primitive précédente mais pour une valeur de barème.

Primitive 5 : SUPPRIMER-DOC-BAR.

Entrée : ID-DOC.

Précond : ID-DOC supposé existant.

Postcond : la documentation qui était identifiée par ID-DOC n'existe plus dans le système.

Objectif : Le but est de supprimer la documentation qui porte sur un barème et qui est identifiée par ID-DOC.

Primitive 6 : SUPPRIMER-DOC-VAL.

Entrée : ID-DOC.

Précond : ID-DOC supposé existant.

Postcond : la documentation qui était identifiée par ID-DOC n'existe plus dans le système.

Objectif : Le but est de supprimer la documentation qui porte sur une valeur de barème et qui est identifiée par ID-DOC.

Primitive 7 : CONSULTER-DOC-BAR.

Entrée : ID-DOC.

Précond : ID-DOC supposé existant.

Sorties : TITRES-DOC, NT, TAB-DOC, LD.

Postcond : les arguments de sorties contiennent des valeurs conformes à l'objectif; NT est compris entre 01 et 10; LD est compris entre 1 et 120.

Objectif : Le but est de fournir dans les arguments de sortie, la documentation identifiée par ID-DOC et qui concerne un barème.

Primitive 8 : CONSULTER-DOC-VAL.

Entrée : ID-DOC.

Précond : ID-DOC supposé existant.

Sorties : TITRES-DOC, NT, TAB-DOC, LD.

Postcond : les arguments de sorties contiennent des valeurs conformes à l'objectif; NT est compris entre 01 et 10; LD est compris entre 1 et 120.

Objectif : Le but est de fournir dans les arguments de sortie, la documentation identifiée par ID-DOC et qui concerne une valeur de barème.

Module de gestion interactive de barèmes.

Voici maintenant la description du premier module du niveau des traitements. Disposant d'outils de gestion de nos données, nous pouvons donc nous pencher sur la réalisation de nos traitements (sur base de ces outils) et ainsi remplir le cahier des charges qui a été déterminé lors de l'analyse fonctionnelle.

Remarquons que dans notre architecture logicielle, selon les besoins du niveau du haut, on peut ajouter sans problème des primitives dans les modules du niveau du bas.

Notre module contient une primitive par traitement à réaliser.

Chaque primitive concerne une des fonctions déterminées dans l'analyse fonctionnelle. Remarquons que les noms de ces primitives sont tous suivis du suffixe "-INT" afin de ne pas les confondre avec les primitives du module BAREME qui s'occupent de la gestion des données et *non* de réaliser des "traitements" comme c'est le cas ici.

Avertissons également le lecteur que dans le programme qui concerne ce module ("GESTION-INTERACTIVE-BAREMES", annexe 2), les noms de ces primitives ne contiennent pas le suffixe précité. Conservant la même politique de réalisation, chacune d'elles possède un numéro identifiant et doit répondre à la spécification qui a été établie.

Précisons que dans ce module, l'interface de communication est ici l'écran. En effet, ce module établit grâce au module E.S un dialogue avec l'utilisateur afin de déterminer ses demandes et de lui fournir les résultats obtenus. Nous invitons donc le lecteur à consulter l'annexe 3 où il y trouvera tous les écrans.

Après avoir exposé l'objectif de ce module, proposons au lecteur de reprendre une à une les neuf primitives du modules pour en exposer leurs noms respectifs et ceux de leurs entrées/sorties. Nous reprendrons en résumé les objectifs de ces primitives car ils ont déjà été développés (avec toutes les contraintes du système qu' ils doivent respectés) dans le chapitre II. Pour la signification des arguments, précisons qu'on utilise ici les mêmes noms que dans les modules précédents. Nous ne les exposerons donc pas une nouvelle fois sauf pour les arguments nouveaux.

L'objectif du module :

L'exécution de ce module commence par l'identification d'un utilisateur et si celle-ci réussit, on poursuit par l'affichage d'un menu constitué de neuf options. Le traitement correspondant à l'option demandée, sera excécuté. Voici donc ce menu :

- 0 / Sortir du programme.
- 1 / Création de barème.
- 2 / Création d'une valeur dans un barème.
- 3 / Obtention des libellés d'une valeur de code.
- 4 / Affichage d'un barème.
- 5 / Obtention de la composition d'un barème.
- 6 / Obtention des noms de tous les barèmes.
- 7 / Recherche selon critère.
- 8 / Obtention des accès autorisés pour l'utilisateur courant.
- 9 / Nettoyage du système.

Voici maintenant la description de nos primitives :

Primitive 1 : CREER-BAREME-INT.

Entrées : ID-BAR, LIBS-BAR, NL, LG-VAL, TP-VAL, DT-CR, DR-RET, NB-VERS, TAB-BARS et NB à lire à l'écran.

Précond : Pour que ces lectures soient acceptées, elles doivent satisfaire les conditions qui suivent.

ID-BAR inexistant; NL compris entre 01 et 10; LIBS-BAR contient NL libellés non existants; LG-VAL est compris entre 01 et 06; TP-VAL = "AN" ou "AB" ou "NU"; DR-RET compris entre 0001 et 9999; NB-VERS compris entre 01 et 20; NB compris entre 00 et 10; TAB-BARS contient NB identificateurs existants de barèmes composants (simples).

Si $NB > 0$ alors $DR-RET = \min (DR-RET \text{ des barèmes composants})$

Sorties : messages de réussite ou d'échec à l'écran.

Postcond : si réussite alors le système contient un barème de plus identifié par ID-BAR.

Objectif : Il s'agit d'essayer de créer un nouvel en-tête de barème dans le système. (Motifs de refus et contraintes à respecter voir analyse fonctionnelle. Idem pour les traitements qui suivent). Cette primitive correspond à la fonction "Création de barème". (cfr 2.4.a.1)

Primitive 2 : CREER-VALEURS-INT.

Entrées : LIBLU, VAL-CODE, LIBS-BAR, NL, DT-CR, DT-DB, HR-DB, DT-FN, TAB-RELS, NV à lire à l'écran.

Précond : Pour que ces lectures soient acceptées, elles doivent satisfaire les conditions qui suivent.

LIBLU contient un libellé de barème existant; VAL-CODE inexistant dans le système et conforme au type et à la longueur maximale spécifiée dans ce barème de nom LIBLU NL compris entre 01 et 10; LIBS-BAR contient NL libellés; NV compris entre 00 et 10; TAB-RELS contient NL valeurs d'actualité des barèmes composants du barème de nom LIBLU; les dates et l'heure doivent être conforme aux contraintes de l'analyse qui portent dessus.

sorties : messages de réussite ou d'échec à l'écran.

Postcond : si réussite alors une nouvelle valeur est créée dans le barème de libellé LIBLU.

Objectif : Essayer de créer une valeur dans le barème de libellé LIBLU.

Cette primitive correspond à la fonction "Création d'une valeur de barème". (cfr 2.4.a.2)

Primitive 3 : OBTENIR-LIBELLES-INT.

Entrées : LIBLU, VAL-CODE à lire à l'écran.

Précond : Pour que ces lectures soient acceptées, elles doivent satisfaire les conditions qui suivent.

LIBLU doit être existant et VAL-CODE non vide.

Sorties : En cas de succès, on obtient à l'écran VAL-CODE, LIBS-BAR, NL, DT-CR, DT-DB, HR-DB, DT-FN et la même chose pour les valeurs de code composantes (dans TAB-RELS de longueur NV) si la valeur de code VAL-CODE est composée. En cas d'échec, un message est affiché à l'écran.

Postcond : En cas de succès, on obtient dans LIBS-BAR (NL sa longueur), TAB-RELS (NV sa longueur), DT-CR, DT-DB, HR-DB,

DT-FN les valeurs qui dans le barème de libellé LIBLU correspondent à VAL-CODE.

Objectif : Obtenir les renseignements qui portent sur VAL-CODE dans le barème de libellé LIBLU. Cette primitive correspond à la fonction "Obtention des libellés d'une valeur de code". (cfr 2.4.a.3)

Primitive 4 : AFFICHER-BAREME-INT.

Entrée : LIBLU à lire à l'écran.

Précond : LIBLU doit être existant.

Sorties : on obtient à l'écran, tout le barème désiré ou un message d'erreur. Le barème se trouve dans les arguments suivant : ID-UTIL, NB-VERS, DR-RET, TP-VAL, LIBS-BAR, NL, LG-VAL, TAB-BARS, NB, TABLEAU-VALS, NLIG, TABLEAU-LIBS.

Postcond : si tout se passe bien, on obtient dans les arguments de sortie, les valeurs qui correspondent au barème de libellé LIBLU.

Objectif : Essayer d'obtenir à l'écran le barème dont le libellé lu en entrée et existant est LIBLU. Précisons qu'on ne désire voir que les valeurs qui y sont d'actualité. Cette primitive correspond à la fonction "Affichage d'un barème". (cfr 2.4.a.4)

Primitive 5 : OBTENIR-COMPOSITION-INT.

Entrée : LIBLU à lire à l'écran.

Précond : LIBLU doit être existant.

Sorties : Messages d'erreur ou une liste de libellés à l'écran.

Postcond : si tout se passe bien, on obtient successivement dans LIBS-BAR les libellés adéquats et conforme à l'objectif.

Objectif : Tenter d'afficher à l'écran la composition du barème dont le libellé est LIBLU c-à-d :

- si le barème LIBLU est simple, les libellés des barèmes qui l'admettent comme composant;
- et s'il est composé, les libellés de ses barèmes composants.

Cette primitive correspond à la fonction "Obtention de la composition d'un barème". (cfr 2.4.a.5)

Primitive 6 : OBTENIR-TOUS-NOMS-INT.

Entrée : /

Sortie : une liste de libellés à l'écran.

Postcond : en cas de succès, LIBS-BAR contiendra successivement les libellés conformément à l'objectif.

Objectif : Sortir à l'écran les libellés de tous les barèmes. Cette primitive correspond à la fonction "Obtention des noms de tous les barèmes". (cfr 2.4.a.6)

Primitive 7 : RECHERCHE-CRITERE-INT

Entrées : LIBLU, TAB-RELS (NV sa longueur) à lire à l'écran.

Précond : Pour que ces lectures soient acceptées, elles doivent satisfaire les conditions qui suivent.

LIBLU doit être existant; TAB-RELS doit contenir NV entrées; chacune de celles-ci doit contenir un identificateur de valeur d'actualité de barèmes composants distincts. De plus LIBLU doit être un libellé de barème "composé".

Sorties : un message d'erreur ou un ou plusieurs tableaux de valeurs qui résultent de la recherche, à l'écran.

Postcond : si tout se passe bien, TABLEAU-LIBS contient les libellés ou titres des colonnes du tableau à sortir, LIBBAR contient le titre du tableau et TABLEAU-VALS contient les valeurs de barèmes qui résultent de cette recherche. Seule, les valeurs d'actualité sont désirées.

Objectif : Il s'agit d'effectuer une recherche selon critère dans le barème de libellé LIBLU avec le critère de recherche lu et valider dans TAB-RELS, et de sortir les différents tableaux résultants à l'écran.

Cette primitive correspond à la fonction "Recherche selon critère". (cfr 2.4.a.7)

Primitive 8 : OBTENIR-ACCES-INT.

Entrées : /

Sorties : soit un message (Aucun accès) ou une liste des libellés des barèmes auxquels l'utilisateur courant (connu par son ID-BAR) a accès.

Postcond : s'il y a des accès, LIBS-BAR (longueur NL) contiendra successivement les libellés des barèmes qui concernent ces accès.

Objectif : Sortir à l'écran les libellés des barèmes auxquels l'utilisateur courant a accès. Cette primitive correspond à la fonction "Obtention des accès autorisés". (cfr 2.4.a.8)

Primitive 9 : NETTOYAGE-SYSTEME-INT.

Entrée : /

Sorties : Une liste de libellés des barèmes qui ont été supprimés du système. A chaque libellé est associé l'identificateur de barème correspondant. Soit cette liste est affichée à l'écran, soit c'est un message prévenant qu'aucun barème n'a été supprimé.

Postcond : si des barèmes sont supprimés, TABLEAU-LIBS contiendra la liste spécifiée ci-dessus.

Objectif : Supprimer du système toutes les valeurs périmées (en commençant par les barèmes composés), tous les barèmes n'ayant plus de valeur et sortir à l'écran la liste de ceux-ci. Cette primitive correspond à la fonction "Nettoyage du système". (cfr 2.4.a.9)

Module de gestion de barème par programmes.

Ce module est donc constitué de deux primitives correspondant aux traitements 3 et 7 du module précédent. La différence est qu'ici tout s'exécute sans interaction à l'écran. Le module reçoit les demandes et les entrées par les arguments de son interface qui auront été garnis par le programme appelant.

Ces primitives ne sont pas à confondre avec celles du module BAREME. Afin de les différencier, on a ajouté à leurs noms le suffixe "-PROG". Remarquons que ce dernier n'est pas présent dans les noms de primitives du programme qui concerne ce module.

1/ Interface et objectif du module.

L'interface du module est constitué de :

& NOM-UTIL est le nom de l'utilisateur;

& PSWD-UTIL est son mot de passe;

& LIBLU est un libellé de barème;

& OPT-MENU est le numéro du traitement demandé;

& TAB-CRIT est le critère de recherche et LCRIT sa longueur;

& TABLEAU-VALS est un tableau qui contiendra les valeurs résultant d'une recherche selon critère;

& NLIG est la longueur de ce tableau;

& W,CLE,TERMI cfr module BAREME;

& LIBBAR est un libellé de barème;

& TABLEAU-LIBS est un tableau de libellés de barèmes;

& NB est sa longueur;

& NUM-ERR est un numéro d'erreur (détail voir analyse fonct.);

& VAL-CODE est une valeur de code;

& DT-CR est une date de création;

& DT-DB, DT-FN sont les dates de début et de fin d'utilisation;

& HR-DB est une heure de début d'utilisation;
 & LIBS-BAR est un tableau de libellés de barèmes, NL sa longueur;
 & TAB-RELS est un tableau d'identificateurs de valeurs de barèmes
 composantes et NV sa longueur.

L'objectif du module :

Le but de ce module est celui-ci. Si OPT-MENU vaut 1, il s'agit d'exécuter le traitement qui consiste à effectuer une recherche selon critère et si c'est 2, il s'agit de celui qui consiste à rechercher les renseignements qui concernent une valeur de code dans un barème.

2/ Description des primitives.

Primitive 1 : RECHERCHE-CRITERE-PROG

Entrées : NOM-UTIL,PSWD-UTIL,OPT-MENU, LIBLU, TAB-CRIT ET
 LCRIT.

Précond : Aucune condition sur les entrées. Le système
 doit être robuste.

Sorties : TABLEAU-VALS,NLIG,W,CLE,TERMI,LIBBAR,NUM-ERR,NB,
 TABLEAU-LIBS.

Postcond : Même chose que le module précédent (primitive 7)
 sauf que l'on ne renvoie qu'un tableau de va-
 leurs à chaque appel de cette primitive. Si W=1
 alors il s'agit du premier tableau sinon du tab-
 leau des valeurs qui suivent la valeur de CLE.

Objectif : Si l'utilisateur est reconnu, si le barème spéci-
 fié par LIBLU existe et est simple, et que le
 critère de recherche est correct, la recherche
 selon critère est exécutée et les résultats sont
 mis dans les arguments de sortie. Quand la main

est rendue au programme appelant, la recherche n'est pas terminée si la valeur de TERMI est toujours "0" et pour obtenir la suite, ce programme devra faire un nouvel appel au système jusqu'à ce que TERMI vaille "1". En cas d'erreur NUM-ERR est garni d'un numéro adéquat. Cette primitive correspond à la fonction "Recherche selon critère" mais par programmes. (cfr 2.4.b.1)

Primitive 2 : OBTENIR-LIBELLES-PROG.

Entrées : NOM-UTIL, PSWD-UTIL, OPT-MENU, LIBLU, VAL-CODE.

Précond : Aucune condition sur les entrées.

Sorties : DT-CR, DT-DB, HR-DB, DT-FN, LIBS-BAR, NL, TAB-RELS, NV, NUM-ERR.

Postcond : Même chose que le module précédent (primitive 3) mais en plus, en cas d'erreur NUM-ERR contient un numéro approprié. (Voir liste analyse fonct.)

Objectif : Si l'utilisateur est reconnu, si le barème spécifié par LIBLU est existant, et que la valeur de code donnée est existante dans ce barème, on obtiendra les renseignements qui concernent cette valeur de code dans ce barème sinon NUM-ERR contiendra le numéro d'erreur approprié. Cette primitive correspond à la fonction "Obtention des libellés d'une valeur de code" mais par programmes. (cfr 2.4.b.2)

CHAPITRE IV

Les performances

Ce petit chapitre a pour but d'analyser les performances des traitements "par programmes".

Rappelons-nous que deux des traitements de notre système devaient pouvoir être appelés par des programmes d'application. Ces traitements ne pouvaient donc pas "bloqués" longtemps le programme qui les appelle. Là était le problème lié à la performance.

Examinons tout d'abord le premier des traitements qu'on a baptisé : "Obtention des libellés d'une valeur de code".

Possédant l'identificateur d'un barème et la valeur de code dont on désire les renseignements, cette recherche ne nécessite :

- qu'un seul accès pour retrouver cette valeur;
- et si elle existe :

- * N accès pour obtenir ses libellés ($1 \leq N \leq 10$);

- * M accès pour obtenir les identificateurs de ses valeurs composantes (si la valeur cherchée est composée).

Constatons simplement que l'on obtient immédiatement les renseignements désirés.

Remarquons que ce premier traitement, comme le deuxième, quand ils sont déclenchés par programmes doivent s'occuper de vérifier leurs arguments d'entrée. Si on procède interactivement, ces vérifications sont déjà effectuées au moment de procéder aux accès qui réalisent la recherche adéquate.

Examinons maintenant le traitement fondamental (du point de vue nombre d'accès) du système, c-à-d celui qu'on a baptisé : "La recherche selon critère". Rappelons qu'il s'agit de rechercher, dans un barème composé, toutes les lignes de celui-ci qui contiennent les valeurs de code composantes incluses dans le critère de recherche.

On aurait pu prendre la politique de passer en revue toutes les lignes du barème et de vérifier si elles correspondent au critère donné.

Quand on sait qu'un barème peut contenir de très nombreuses valeurs (lignes), cette solution ne pouvait être performante.

Voici un exemple de barème qui a pour but de faire comprendre au lecteur la solution qui a été implémentée.

Barème des taux d'intérêts à appliquer en compte (IS0004) :

Code ----	Type de comptes -----	Taux d'intérêts -----	Code taux -----
1	CV	3 %	0
2	CD	3 %	0
3	CD	6 %	2
4	CT	9 %	3
	(IS0001)	(IS0002)	(IS0003)

Critère de recherche : obtenir toutes les lignes dont le taux d'intérêts est 3 % et le code taux 1. En d'autres termes, il s'agit de la valeur de code "3 %" (dans le barème d'identificateur "IS0002") et de la valeur de code "1" (dans le barème d'identificateur "IS0003").

Afin de ne pas passer en revue toutes les lignes, et sachant qu'une valeur de barème n'est représentée qu'une seule fois dans le système, la solution implémentée part de la première valeur composante du critère. Il faut donc partir de la valeur de code "3 %" du barème des taux d'intérêts et de remonter aux lignes (valeurs) des barèmes composés auxquelles elle est reliée. (1 accès)

Chaque fois qu'on en obtient une, les questions suivantes doivent ensuite se poser : la ligne trouvée :

- appartient-elle bien au barème, objet de la recherche ? (1 accès)
- est-elle d'actualité dans ce barème ? (0 accès, c'est un calcul de date)

- est-elle reliée aux autres valeurs de code composantes ?

(c-à-d "1" du barème des codes taux) (N accès, $1 \leq N \leq 10$)

Si les réponses sont affirmatives, la ligne est une partie de la solution.

Cette dernière a le désavantage de passer en revue des lignes qui ne font pas partie du barème désiré. Mais cela est plus avantageux que de parcourir séquentiellement plusieurs centaines de lignes de barème.

Une amélioration possible serait de réaliser dans le système un module qui procède à des calculs statistiques de ce genre :

1 / Quelle est la longueur d'un barème c-à-d quel en est son nombre de valeurs ?

2 / Combien de fois une valeur de simple intervient-elle dans tous les barèmes composés ? (Un chiffre par barème et un chiffre global)

etc.

Ceci permettrait au système de choisir entre plusieurs méthodes de recherche afin d'optimiser au mieux le nombre d'accès.

Exemple : si la longueur du barème est plus petite que X valeurs, il vaut mieux utiliser une méthode séquentielle; si cette longueur est plus grande que Y valeurs, il est préférable d'employer la méthode implémentée tout en choisissant, selon sa fréquence d'apparition dans les barèmes composés, la valeur de code de départ dans notre recherche. Ici on avait choisi aléatoirement la première.

Il en découle qu'il sera sans doute possible d'améliorer le nombre d'accès en disposant de ces chiffres. Dans ce cas, il serait nécessaire d'examiner la possibilité de réaliser ces calculs, certains pouvant se révéler ardues vu la taille du système.

CH A P I T R E V

Conclusions

Conclusion.

Nous voici arrivés à la fin de ce travail. Il est donc temps de conclure.

La première chose à remarquer est que ce système n'est qu'un prototype. Il a pour but d'étudier les possibilités d'obtenir un jour un système performant. Dans ce but, certains choix de réalisation ont été entrepris dans l'espoir d'atteindre cette performance. Cela a conduit à ne pas réaliser tout ce qu'on aurait voulu afin de concentrer toute l'énergie sur le noeud du problème.

La partie la plus importante a donc été implémentée afin de pouvoir répondre à la question : " Aura-t'on un jour un système performant ?"

Ce prototype n'est en aucune manière un idéal, mais sa réalisation a finalement permis de proposer certaines améliorations futures possibles (cfr chapitre IV). Là était le but de cette étude.

Au sujet des deux traitements dont on visait la performance (vu leurs appels par programmes), on a pu faire les constatations suivantes: l'obtention des libellés (dates et heure) d'une valeur de code n'a pas posé véritablement de problèmes; quant à la "recherche selon critère", qui était la tâche la plus délicate (le système peut en effet être d'un certain volume), le choix de la réalisation qui a été fait a eu pour objet de limiter le nombre d'accès dans le système. L'algorithme fondamental qui concerne ce traitement peut être consulté à l'annexe 1 (module BAREME), son explication et ses améliorations au chapitre IV.

Remarquons également que, dans ce système, on a exigé une certaine sécurité. On ne peut donc s'y introduire comme on le désire et logiquement, celui-ci ne peut faire confiance ni à l'utilisateur, ni à un programme appelant.

Bien sûr, quand on exige la sécurité dans un système, elle joue souvent en défaveur de la performance.

En vue de la réalisation d'un système opérationnel, on pourrait donc conseiller de continuer l'étude de l'accès aux données et de la sécurité du système, en étant bien conscient de son action néfaste sur la performance, de ne pas négliger la gestion de la documentation et bien sûr de penser à réaliser le module ES à l'aide d'un gestionnaire d'écran permettant d'améliorer l'aspect communication avec l'utilisateur. Il reste donc encore beaucoup de chemin à parcourir...

Bibliographie

Bibliographie.

- Conception des applications informatiques
1. étude d'opportunité et analyse conceptuelle
F.BOBART Y.PIGNEUR (Edition MASSON)
- Méthodologie de développement de logiciels.
Cours de deuxième licence et maîtrise en informatique.
A. VAN-LANSWEERDE
- Système de gestion de base de données.
Cours de première licence et maîtrise en informatique.
J-L. HAINAUT
- Analyse des accès proposés par le DBTG CODASYL (1971)
J-L. HAINAUT
- Cadre de référence pour la conception de base de données.
J-L. HAINAUT

Implantation d'une gestion

de barèmes

LES ANNEXES.

E. Evrard
F.U.N.D.P.
3e Licence et maîtrise
en Informatique
Année académique 1985-1986

Promoteur R. Lesuisse
F.U.N.D.P.

Responsable extérieur P. Taton
Crédit Communal de Belgique

LES ANNEXES .

ANNEXE 1 :

Les algorithmes principaux .

1/ Module BAREME .

1. Primitive CR-TETE-BAR.

Ouvrir la B.D;

Mettre ID-BAR dans l'item NR-BAR de l'article BAREME ;

Mettre DT-ER dans l'item DATE-CREATION " " ;

" TP-VAL " " TYPE-VAL " " ;

" LG-VAL " " LONG-VAL " " ;

" NB-VERS " " NBR-VERS-GARD " " ;

" DR-RET " " DUR-RETENTION " " ;

Enregistrer cet article BAREME ;

$I = 1$;

répéter

Mettre LIBELLE (I) dans l'item VAL-LIB-BAR de l'article LIB-BAR ;

" REG-LING (I) " " CODE-AL-BAR " " ;

Enregistrer cet article ; (insertion automatique dans le chemin LIEN-BAR-LIB)

$I = I + 1$;

Jusqu'à ce que $I > NL$

$I = 1$

Tant que $I \leq NB$

Mettre IDENT-BAR (I) dans l'item NR-BAR-LB de l'article LIEN-BAR ;

" ROLE-BAR (I) " " ROLE-LB " " ;

Enregistrer cet article LIEN-BAR ;

(Insertion automatique dans le chemin EST-COMP-DE)

$I = I + 1$

Fermer la B.D.

3. Primitive EXIST-BAR.

Ouvrir la B.D.

Accès à l'article BAREME dont la clé NR.BAR vaut ID.BAR.

alors

Si on le trouve

sinon

EXISTE = 1

EXISTE = 0

Fermer la B.D.

4. Primitive OBTENIR-ALL-IDBAR.

ouvrir la B.D.

$FINI = 0$; $I = 1$

Alors

Passer à l'article BAREME suivant.

s'il y en a un

alors

sinon

Mettre l'item NR.BAR dans
 $NBAR(I)$

$FINI = 1$

$I = I + 1$

jusqu'à ce que $FINI = 1$

$NC = I - 2$

Fermer la B.D.

5. Primitive EXIST-LIB-BAR.

ouvrir la B.D.

Accès à l'article LIB-BAR dont la clé VAL-LIB-BAR vaut LIBLU

Si on le trouve

alors

sinon.

EXISTE = 1

Remonter à l'ouvrir BAREME
du chemin LIEN-BAR-LIB et
dont le membre est cet article
LIB-BAR trouvé.

EXISTE = 0

Mettre la clé NR-BAR de
cet article BAREME dans
ID-BAR.

Fermer la B.D.

6 Primitive OBTENIR-COMPOSE.

Ouvrir la B.D.

$FINI = 0$; $I = 1$;

Répéter

Si $I > 1$ alors retour au courant qui est mémorisé ;

accès à l'article LIEN-BAR suivant dont la valeur de GENR-BAR (Barant ID-BAR

Si on en trouve encore

alors

sinon

Mémoriser son courant.

$FINI = 1$

Remonter à son owner BAREME
dans le chemin EST-EMP-DE

Mettre l'item NR-BAR de BAREME
dans NBAR(I)

$I = I + 1$

Jusqu'à ce que $FINI = 1$

$NC = I - 2$

Fermer la B.D.

7. Primitive CR-VAL-BAR.

Ouvrir la B.D.;

Mettre ID.BAR dans NRBAR-REL de l'article REL-BAR-CODE.

" ID-REL " NRBAR-VAL-CODE " " "

" DT-CR " DATE-CREATION " " "

" DT-DB " DATE-DEB-UTIL " " "

" HR-DB " HEURE-DEB-UTIL " " "

" DT-FN " DATE-FIN-UTIL " " "

Enregistrer cet article REL-BAR-CODE

I = 1

Répéter

Mettre LIBELLE (I) dans l'item VAL-LIB-LVT de l'article LIB-VAL;

" REG-LING(I) " " CODE-RE-LVT " " " " ;

Enregistrer cet article LIB-VAL; (insertion automatique dans le chemin
LIEN-REL-LIB)

I = I + 1

Jusqu'à ce que I > NV

I = 1

Tant que I ≤ NV

Mettre NREL (I) dans l'item REL-COMP-LR de l'article LIEN-REL;

" RREL (I) " " ROLE-LR " " " " ;

Enregistrer cet article LIEN-REL;

(Insertion automatique dans le chemin
LIEN-RBC)

I = I + 1

Fermer la B.D.

8. Primitive OBTENIR_VAL_BAR.

Ouvrir B.D.

Associer à l'article REL-BAR.CODE dont l'adresse NABAR.VALCODE vaut ID-REL.

alors

Si on le trouve

sinon

EXISTE = 1

Mettre l'item DATE-CREATION de l'article dans DT-CR

" " DATE-DEB-UTIL " " DT-DB

" " HEURE-DEB-UTIL " " HR-DB

" " DATE-FIN-UTIL " " DT-FN

FINI = 0 ; I = 1 ;

Repetér

accéder à l'article LIB.VALCODE suivant dans le chemin LIEN.REL-LIB

Si il y en a un

alors Mettre l'item VAL-LIB-LVT dans LIBELLE(I)

" " CODE-AL-LVT " " REG-LING(I)

sinon FINI = 1

I = I + 1

jusqu'à ce que FINI = 1

NL = I - 2 ; FINI = 0 ; I = 1 ;

Repetér

accéder à l'article LIEN-REL suivant dans le chemin LIEN-ABE

Si il y en a un

alors Mettre l'item REL-COMP-LR dans NREL(I)

" " ROLE-LR " " RREL(I)

sinon FINI = 1

I = I + 1

jusqu'à ce que FINI = 1

NV = I - 2

Fermer B.D.

EXISTE = 0

9 Primitive PAS-DE-BAREMES.

Ouvrir la B.D.

Accès au 1^{er} article BAREME.

Si on le trouve

alors

sinon

EXISTE = 1

(Il y a au moins un
barème dans le système)

EXISTE = 0

(Il n'y a aucun
barème dans le système)

Fermer la B.D.

10 Primitive PAS-DE-VALEURS.

ouvrir la B.D.

Accès au 1^{er} article REL-BAR.CODE dans le barème identifié par ID-BAR.

Si on le trouve

alors

sinon

EXISTE = 1

(Il y a au moins une valeur dans le barème)

EXISTE = 0

(Il n'y a pas de valeur dans le barème)

Fermer la B.D.

11 Primitive GIVE_HT-VALS.

Ouvrir la B.D

STOPPE = 0 ; J = 1

Répter

Trouver dans le barème identifié par ID-BAR, l'article REL-BAR-CODE suivant

Si cette valeur REL-BAR-CODE existe

alors si elle est d'actualité

alors la placer avec ses valeurs composantes
dans TABLEAU-VALS

sinon J = J + 1

sinon STOPPE = 1 ; TERM1 = 1 ; NLIG = J - 1

Si J = 8

alors STOPPE = 1 ; NLIG = 8

fini

J = J + 1

Jusqu'à ce que STOPPE = 1

Si TERM1 = 0

alors ELE = identifiant de REL-BAR-CODE

sinon ELE = " "

Fermer la B.D.

12 Primitive RECH-HT-VALS

Ouvrir la B.D.

Si $W > 1$ alors retour à l'environnement d'appel précédent (grâce à @LE)
et NREL(1)

$I = 1$

Boucler

FINI = 0

Boucler

Accès à l'article LIEN-REL suivant dont REL-COMP-LR = NREL(1) c'est-à-dire accès à un lien vers la valeur composante NREL(1)

Si cet article LIEN-REL existe

alors - mémoriser son courant*;

- remonter à l'obtenir REL-BAR-CODE du chemin LIEN-RBC dont cet article est membre;

- Si cette valeur REL-BAR-CODE \in Barème identifié par ID-BAR

alors si cette valeur REL-BAR-CODE est d'actualité

alors Mettre son identifiant dans ELE;
si elle est reliée aux autres valeurs
composantes du critère de recherche
c'est-à-dire (NREL(2) \rightarrow NREL(NV))

alors la placer avec ses valeurs
composantes dans TABLEAU-VALS;
 $I = I + 1$; FINI = 1; **

sinon revenir au courant mémorisé*

sinon revenir au courant mémorisé*

sinon revenir au courant mémorisé*

sinon FINI = 1; TERMI = 1;
NLIG = I - 1

** une bonne valeur
est trouvée

Jusqu'à ce que FINI = 1 (c'est-à-dire si on a trouvé une bonne valeur)

si $I = 8$ alors NLIG = 8

Jusqu'à ce que $I = 8$ ou que TERMI = 1 (on veut obtenir au plus 8 bonnes valeurs)

Fermer la B.D.

13 Primitive SUPPRESSION_VALEURS.

ouvrir la B.D ; FINI = 0 ;

Répter

STOPPE = 0

Répter

Trouver dans le barème identifié par ID-BAR, la valeur suivante
REL-BAR-CODE

Si elle existe

alors si elle n'est plus d'actualité

alors STOPPE = 1

fini

sinon STOPPE = 1, FINI = 1

Jusqu'à ce que Stoppe soit = 1

Si FINI = 0

alors si cette valeur est périmée

alors supprimer cette valeur
REL-BAR-CODE

fini

fini

↓
(cela entraîne la suppression de tous les
chemins et articles de ces chemins.
Occurrences de chemins visées :

* LIEN-RBC
* LIEN-REL-LIB)

Jusqu'à ce que FINI soit = 1

fermer la B.D (MAS)

14 Primitive SUPPRESSION BAREMES.

Ouvrir la B.D.

Accès à l'article BAREME dont la clé NR-BAR vaut ID-BAR

Supprimer cet article.

(Cela entraîne la suppression de tous les chemins et articles
de ces chemins dont il est owner.

Occurrences de chemins visées : EST-COMP.DE et LIEN-BAR-LIB)

Fermer la B.D.

2/ Module SECURITE .

1. Primitive EXIST-UTILISATEUR

Ouvrir la B.D.

accès à l'article UTILISATEUR dont la clé NR-UTIL vaut ID-UTIL

Si on le trouve

alors

sinon

EXISTE = 1

(l'utilisateur existe dans
le système)

EXISTE = 0

(l'utilisateur n'existe
pas dans le système)

Fermer la B.D.

2. Primitive EXIST_ACCES

ouvrir la B.D.

accès à l'article UTILISATEUR dont la clé NR-UTIL vaut ID-UTIL

$FINI = 0$; $EXISTE = 0$;

Répéter

accès à l'article BAREME existant dans la chemin LIEN_ACCES_BAR.
(lien auteur)

Si on le trouve

alors

sinon

Si l'item NR-BAR
= ID-BAR

alors

sinon

$EXISTE = 1$

$FINI = 1$

$FINI = 1$.

jusqu'à ce que $FINI = 1$.

Fermer la B.D.

3. Primitive CREER-ACCES

Ouvrir la B.D.

accès à l'article BAREME dont la clé NR-BAR vaut ID-BAR

accès à l'article UTILISATEUR dont la clé NR-UTIL vaut ID-UTIL

Insérer l'article UTILISATEUR trouvé dans le chemin
LIEN-ACCES-BAR (lien auteur) dont l'article BAREME
TROUVE' en est l'owner.

Fermer la B.D.

4 Primitive GIVE-ACCES.

ouvrir la B.D.

accès à l'article UTILISATEUR dont le NR.UTIL vaut ID-UTIL .

$\text{FINI} = 0$; $I = 1$;

Débuter

accès à l'article BAREME suivant dans le chemin LIEN-ACCES-BAR
(coteur)

si on le trouve

alors

sinon

Mettre l'item NR-BAR dans
 $\text{NR}(I)$
(5^e composante de TAB)

$\text{FINI} = 1$

$N = I - 1$

$I = I + 1$

Jusqu'à ce que $\text{FINI} = 1$

Fermer la B.D.

3/ Module gestion interactive

de barèmes.

1 Primitive CREER-BAREME

Lire un identificateur de bareme ID-BAR à l'écran jusqu'à ce qu'il soit unique dans le système

Mettre dans DT-PR la date du système } appl module ES
Lire et valider TP-VAL, LG-VAL, NB-VERS, NL } primitive n°5

Pour I = 1 jusqu'à NL

Lire et valider (LIBELLEI), un libellé de bareme (appl ES) primitive n°10

Si il existe déjà dans le système (appl module BAREME primitive n°5)
alors I = I - 1, + refus avec message

fin

Lire et valider NBP (nombre de baremes composants) ($1 \leq NBP \leq 10$) (appl Mod ES)

Si NBP > 0

alors

sinon

Si il n'existe aucun bareme alors message + retour menu
d'erreur

Pour I = 1 jusqu'à NBP

Lire le libellé d'un bareme composant (appl Mod ES)

Si l'utilisateur n'a pas accès au bareme désigné
alors message + retour menu

Si le bareme désigné n'existe pas ou qu'il est composé, ou qu'il avait déjà été désigné
alors refus avec message ; I = I - 1

Si c'est le 3^e refus d'affiler
alors message et retour menu

SUITE → page suivante

1 Primitive CREER_BAREME (suite)

$NB = NBP.$

Si $NB = 0$

alors lire et valider DR-RET (appel module ES)

Si $NB > 0$

alors $DR-RET = \min \{ DR-RET \text{ des baremes composants } \text{de bareme} \}$

Appel au module BAREME afin d'y stocker

notre bareme (primitive n°1) et SECURITE (primitive n°4)
pour en avoir un accès à ce bareme!

2. PRIMITIVE CREER-VALEURS.

Si il n'existe aucun barème dans le système alors message + retour menu
(appel module BARREME)

Lecture d'un libellé de barème. (appel module ES)

Si ce barème n'existe pas alors message + retour menu
sinon on obtient l'en-tête de ce barème dans les variables

Si l'utilisateur n'a pas accès à ce barème alors message + retour menu
(appel module SECURITE)

Si un des barèmes composants ne contient pas de valeurs alors fin + retour menu
(appel module BARREME)

répéter

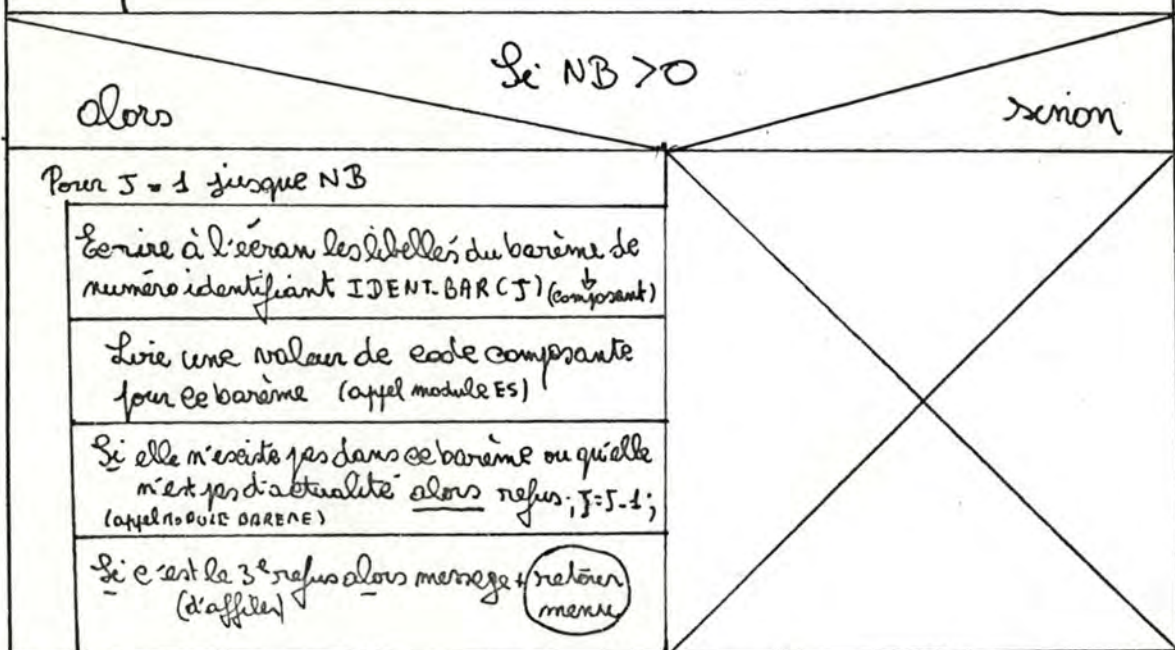
Lecture et validation de VAL.CODE (une valeur de code)

Jusqu'à ce que VAL.CODE soit non vide, insérée dans ce barème, comptable avec le type spécifié dans TP.VAL et avec la longueur maximum spécifiée dans LG.VAL

Lecture et validation de NL (nombre de libellés)

Pour I = 1 jusqu'à NL

Lecture et validation de LIBELLE(I)



SUITE → Page suivante

2. Primitive CREER-VALEURS (suite)

$NB = NV$

Si $NV = 0$

alors $DT-ER = \text{date du système};$

Lire et valider $DT-DB$ jusqu'à ce que $DT-ER \leq DT-DB$;

Lire et valider $HR-DB$ (avec le contrôle : si $DT-ER = DT-DB$
alors $HR-DB \geq \text{heure système}$);

Lire et valider $DT-FN$ jusqu'à ce que $DT-DB \leq DT-FN$;

Si $NV > 0$

alors $DT-ER = \text{date du système};$

$DT-DB = \text{date du système};$

$HR-DB = \text{heure du système};$

$DT-FN = \min \{ DT-FN \text{ des valeurs composantes} \}$

Appel au module BAREME pour y stocker la nouvelle
valeur de bareme. (primitive n°2)

3. PrIMITIVE OBTENIR_LIBELLES_VALEUR.

S'il n'existe aucun barème alors message + retour menu.
(appel module BAREME)

Lecture d'un libellé de barème (LIBLU). (appel module ES)

Si ce barème n'existe pas alors message + retour menu.
sinon détermination du ID-BAR (Nat. BAR Primit n° 5)

Si l'utilisateur n'a pas accès au barème alors message + retour menu.
(appel module SECURITE)

S'il n'y a pas de valeur dans ce barème alors message + retour menu.

Lecture d'une valeur de code (VAL-CODE) + constitution de ID-REL (appel ES)

Appel au module BAREME primitive n° 8 pour obtenir les libellés
dates et l'heure de cette valeur de code identifiée par ID-REL

Si EXISTE = 1	
alors	sinon
Sortir à l'écran VAL-CODE, DT.ER, DT.DB, HR.DB, DT.FN et LIBS-BAR de longueur NL	message + <u>retour menu</u>
Pour I = 1 jusqu'à NV	
appel au module BAREME (primitive n° 2) pour obtenir les mêmes renseignements pour la valeur de code composante contenue dans NREL(I)	
Sortir à l'écran ces renseignements.	

4 Primitive AFFICHER-BAREME.

Si aucun barème dans le système alors message + retour menu (appel mod BAREME)

Lecture d'un libellé de barème. (appel module ES) (Variable LIBLU)

Si le barème n'existe pas alors message + retour menu (appel mod BAREME) (NB BAR primit n°3)

Si l'utilisateur n'a pas accès à ce barème alors message + retour menu (appel module SECURITE)

Appel au module BAREME pour obtenir l'en-tête de ce barème par le ID-BAR obtenu lors de l'opération précédente. (primitive n°2)

LIBBAR = LIBELLE(1)

Si NB > 0 alors mettre dans TABLEAU-LIBS les libellés des barèmes composants

Sortir et en-tête à l'écran et à ID-BAR, LIBS-BAR de longueur NL, TP-VAL, LG-VAL, DR-RET, NB-VERS, DT-ER et nom du créateur.

Demander si l'utilisateur désire voir les en-têtes des barèmes composants.

alors

Si la réponse est "0"

sinon

pour Z = 1 jusqu'à NB

accès à l'en-tête du barème composant de numéro IDENT-BAR(Z)

La sortir à l'écran.

Si il existe des valeurs dans ce barème LIBLU

alors

sinon

accès et impression de celles-ci pour groupe de 8 et sous forme de tableau.

* accès → appel au module BAREME Primitive n°11 ;

* impression → appel au module ES Primitive n°24 et n°25.

Message + retour menu

5. Primitive OBTENIR.COMPOSITION.

Si il n'existe aucun barème alors message + retour menu
(appel module BAREME)

Lecture d'un libellé de barème. (appel module ES)

Si le barème n'existe pas alors message + retour menu
sinon obtention du ID-BAR (Mod BAR-Prim³)

Si l'utilisateur n'a pas accès à ce barème alors message + retour menu
(appel module SECURITE)

Appel au Module BAREME (primitive n°2) pour obtenir l'en-tête de ce barème.

Sortir à l'écran les libellés de ce barème (LIBS-BAR, de longueur NL)

alors

Si NB > 0

sinon

Pour J = 1 jusqu'à NB

Accès à l'en-tête du barème de N°
IDENT-BAR(J) (appel module BAREME)

Sortir les libellés de ce barème à
l'écran. (appel module ES)

alors

Si NB = 0

sinon

appel au module BAREME (primitive n°6) pour
obtenir les identifiants de barèmes qui
admettent ce barème identifié par ID-BAR
comme composant.

Pour J = 1 jusqu'à NE

Accès à l'en-tête du barème de N°
NBAR(J) (appel module BAREME)

Sortir les libellés de ce barème
à l'écran. (appel module ES)

6. Primitive OBTENIR_TOUS_NOMS.

Si il n'existe aucun barème alors message + (retour menu)

Appel au module BAREME (primitive n°4) pour obtenir tous les identifiants des barèmes du système (dans TAB-IDS, longueur NE)

FINI = 0 ; Z = 0 (compteur du nbr de barèmes)

répéter

EC = 0 (indicateur de groupe de 4) ; J = 1 (indice de boucle)

Répéter

Z = Z + 1

Si J = 4 alors EC = 1

Si Z = NE alors FINI = 1 ; EC = 1 ;

Appel au module BAREME pour obtenir l'en-tête du barème N° NBAR(Z)

Sortir les libellés de ce barème à l'écran (appeler module ES)

J = J + 1

Jusqu'à ce que EC = 1

Demander si on veut continuer ou s'arrêter ?

Si réponse est "S" alors (retour menu)

Jusqu'à ce que FINI = 1

Effacer le tableau TAB-IDS.

7. Primitive RECHERCHE_CRITERE.

Si pas de barème alors message + retour menu (appel Mod. BAREME P n° 9)

Lire un libellé de barème. (LIBELU) (appel Mod ES)

Si le barème désigné n'existe pas alors message + retour menu (appel Mod BAREME P n° 5)
si non obtention du ID-BAR

Si l'utilisateur n'a pas accès à ce barème alors message + retour menu
 (appel mod SECURITE)

Si il n'existe aucune valeur dans ce barème alors message + retour menu
 (appel mod BAREME P n° 10)

Si le barème désigné est simple alors message + retour menu nombre de barèmes composants
 (Pour le savoir, on fait un appel au Module BAREME primitive 2 et on regarde la NB)

NBP = 1

Pour $S = 1$ jusqu'à NB

appel au module BAREME primitive 2 (sur 2^e liste de variables) pour obtenir
 l'en-tête du barème de n° IDENT-BAR (S)

Afficher les libellés de ce barème composant à l'écran.

Demander s'il fait partie de la recherche (lire une réponse jusqu'à ce que
 la valeur de celle-ci soit "O" ou "N")

Si réponse = "O"

alors $\left\{ \begin{array}{l} \text{mémoiser l'identificateur de barème car il fait partie de la recherche} \\ \text{c'est IDENT-BAR}(NBP) = \text{IDENT-BAR}(S); \\ NBP = NBP + 1; \\ \text{LIBEL}(S) = \text{LIBELLE}(1); \text{ (ici on met le 1^{er} libellé de ce barème dans} \\ \text{TABLEAU-LIBS)} \end{array} \right.$

Si $S = NB$ et $NBP = 0$ (c'est à dire qu'on a passé tous les barèmes composants en revue
 et qu'aucun d'eux ne fait partie de la recherche)

alors message (pas de critères) + retour menu

Suite → page suivante

7. Primitive RECHERCHE CRITERE (suite)

$NV = NBP$

Pour $S = 1$ jusqu'à NV

appel au module BAREME primitive 2 pour obtenir l'en-tête du barème
no IDENT-BARP(S)

afficher les libellés de ce barème composant (appel mod ES)

Demander une valeur de code composante. (appel mod ES)

Si la valeur de code entrée n'existe pas dans ce barème composant
alors $S = S - 1$; (+ message à l'écran)

Si cette valeur de code n'est pas d'actualité
alors $S = S - 1$; (+ message à l'écran)

Si c'est le 3^e refus d'affiler
alors message + retour menu

$TERMI = 0$; $W = 1$

Répéter

appel au module BAREME primitive 12 pour procéder à la recherche

Impression à l'écran du tableau trouvé.

Si $W = 1$ et $TERMI = 1$ et $NUL = 0$ (1^{er} tableau et la recherche est terminée et la longueur est 0)
alors impression d'un message (pas de résultat)

$W = W + 1$

Jusqu'à ce que $TERMI = 1$

8 Primitive OBTENIR-ACCES.

Si il n'existe encore aucun barème alors message + retour menu

Appel au module SECURITE (primitive n°4) pour obtenir tous les identifiants des barèmes auxquels l'utilisateur a accès (TAB-IDS, longueur NE)

Si $NE > 0$
alors faire

FINI = 0 ; Z = 0 ;

répéter

EE = 0 ; S = 1 ;

répéter

$Z = Z + 1$

Si $S = 4$ alors $EE = 1$

Si $Z = NE$ alors $FINI = 1$; $EE = 1$

Appel au module Barème pour obtenir l'en-tête du barème n° NBAR(Z)

Sortir les libellés de ce barème à l'écran (appel Module ES)

$S = S + 1$

jusqu'à ce que $EE = 1$

Demander si on veut continuer ou s'arrêter ?

si réponse est "S" alors retour menu

Jusqu'à ce que $FINI = 1$

Effacer le tableau TAB-IDS

sinon ^{faire} message (aucun accès) + retour menu

9. Primitive NETTOYAGE-SYSTEME.

Si aucun barème alors message + retour menu

appel au module BAREME primitive 4 pour obtenir tous les identificateurs de barèmes du système (dans TAB-IDS, longueur NE)

Pour I = 1 jusqu'à NE

appel au module BAREME primitive 2 pour obtenir le nombre de barèmes composants NB du barème N° NBAR(I) * (TAB-IDS)

$TNB(I) = NB$ * (TAB-NB = tableau de NB)

Pour I = 1 jusqu'à NE

Si $TNB(I) > 0$ (voir si c'est un barème composé)

alors appel au module BAREME primitive 13 pour y supprimer les valeurs primées du barème de n° NBAR(I)

Pour I = 1 jusqu'à NE

Si $TNB(I) = 0$ (voir si c'est un barème simple)

alors appel au module BAREME primitive 13 pour y supprimer les valeurs primées du barème de n° NBAR(I)

$J = 0$

Pour I = 1 jusqu'à NE

appel au module BAREME primitive 10 pour voir si le barème de n° NBAR(I) contient des valeurs

Si EXISTE = 0 (ne contient pas de valeur)

alors appel module BAREME Pr n° 2; (appel aux libellés mécaniques)

$J = J + 1$;

$NIBR(J) = NBAR(I)$;

$LIBR(J) = LIBELLE(1)$;

} constitution de la liste des barèmes supprimés.

appel au module BAREME Pr n° 14 pour supprimer ce barème;

$LO = J$ (LO est la longueur du tableau des barèmes supprimés)

Sortir à l'écran la liste des barèmes supprimés TAB-LIBS de longueur LO

4/ Module gestion de barèmes

par programmes.

1. Primitive RECHERCHE CRITERE.

Si il n'y a aucun barème alors $NUM_ERR = 12$; retour programme appelant

Si l'utilisateur n'est pas reconnu alors $NUM_ERR = 1$; retour programme appelant
(appel mod SECU)

Si le barème spécifié par LIBLU n'existe pas alors $NUM_ERR = 2$; retour programme appelant
(appel mod BAREME) sinon on obtient ID-BAR

Si l'utilisateur n'a pas accès à ce barème alors $NUM_ERR = 3$; retour programme appelant
(appel mod SECUITE)

Si il n'y a aucune valeur dans ce barème alors $NUM_ERR = 12$; retour programme appelant
(appel mod BAREME)

Si le barème désigné est simple alors $NUM_ERR = 5$; retour programme appelant
(pour cela, appel mod barème Pr 2)

Pour $J = 1$ jusqu'à LCRIT

EXISTE = 0

REPETER

Si TER.BAR(J) = IDENT.BAR(I) alors EXISTE = 1

Jusqu'à & que $J > NB$ ou que EXISTE = 1

Si EXISTE = 0

alors $NUM_ERR = 6$; retour programme appelant

(c'est que le N° du barème traité ici ne fait pas partie des identif. de barèmes composants)

Mettre les lettres du tableau final (pour les colonnes) dans TABLEAU-LIBS; NV = LCRIT

Pour $J = 1$ jusqu'à NV

Contrôler si la valeur de code TER.VAL(J) existe dans le barème N° TER.BAR(J). (appel mod BAREME)

Si ce n'est pas le cas alors $NUM_ERR = 9$; retour programme appelant

Contrôler si la valeur de code TER.VAL(J) est d'actualité dans le barème N° TER.BAR(J)

Si ce n'est pas le cas alors $NUM_ERR = 7$ ou 8; retour programme appelant

Mettre cette partie de critères dans NREL(J) (TAB.RELS)

Si $W = 00$ alors $W = 01$ sinon $W = W + 1$

TERMI = 0

appel au module BAREME primitive 12 pour procéder à la recherche

Retour au programme appelant

2. Primitive OBTENIR_LIBELLES.

Si l'utilisateur n'a pas été reconnu alors $NUM.ERR = 1$; retour prog appelant
(appel Mod SECURITE)

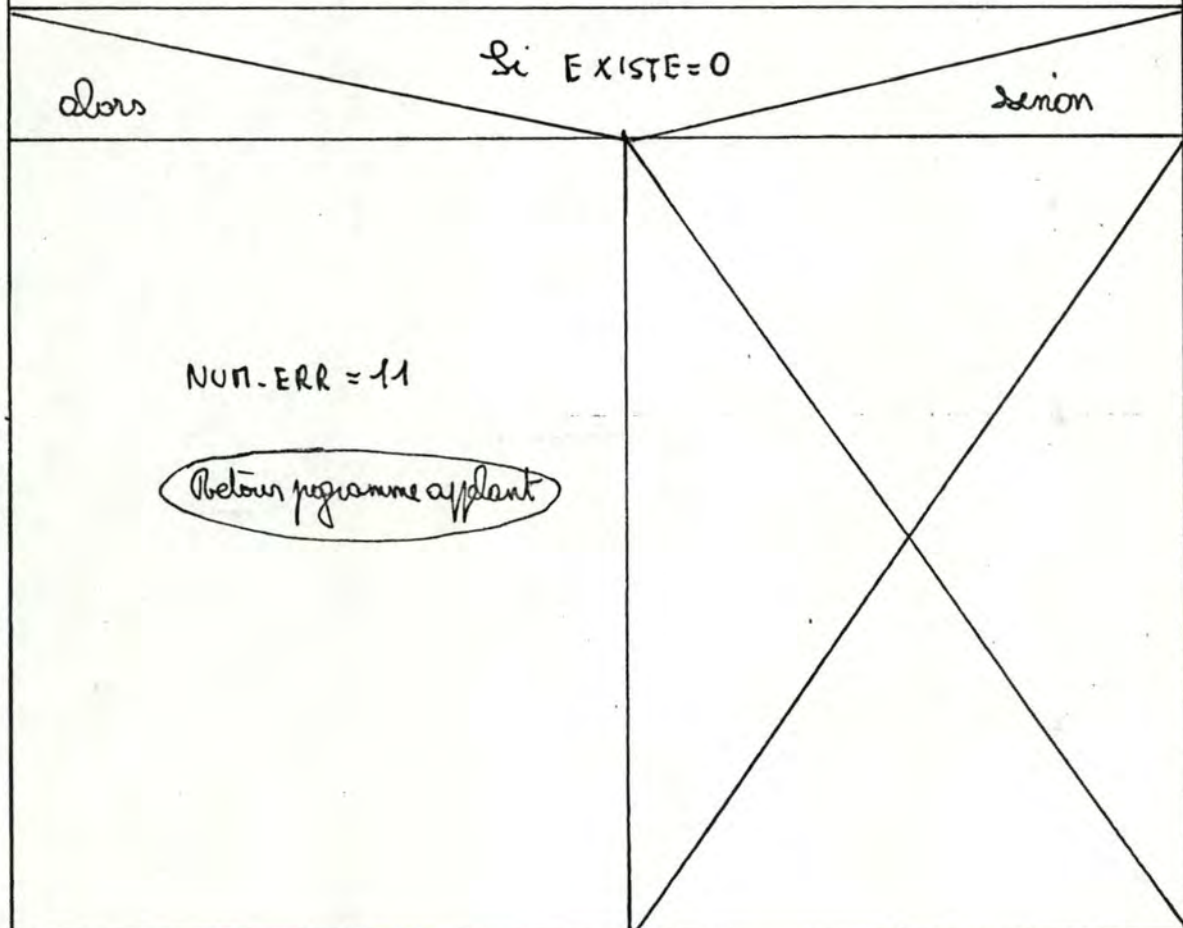
Si il n'existe aucun barème alors $NUM.ERR = 12$; retour programme appelant
(appel Mod BAREME)

Si le barème spécifié par LIBU est inexistant alors $NUM.ERR = 2$ retour prog appelant
(appel Mod BAREME) sinon on obtient son ID-BAR

Si l'utilisateur n'a pas accès alors $NUM.ERR = 3$; retour prog appelant
(appel Mod SECURITE)

Si il n'y a pas de valeur dans ce barème alors $NUM.ERR = 4$; retour prog appelant
(appel Mod BAREME)

Appel au module BAREME primitive 8 pour obtenir les libellés, dates
et heure de la valeur de code donné en entrée VAL.CODE.



ANNEXE 2 :

Les programmes commentés.

Architecture physique de programmation.

Cette architecture se compose de onze fichiers répartis en deux niveaux respectant notre architecture logique. Le niveau du bas regroupe les fichiers qui constituent les modules de données. En voici le détail:

- BAREME.COB pour le module BAREME;
- SECURITE.COB pour le module SECURITE;
- ES.COB pour le module Entrées/Sorties;

Ensuite nous avons six petits fichiers qui ont pour but de réaliser des opérations sur les dates et heures afin de décharger de ces problèmes la programmation des modules .

Ces fichiers sont:

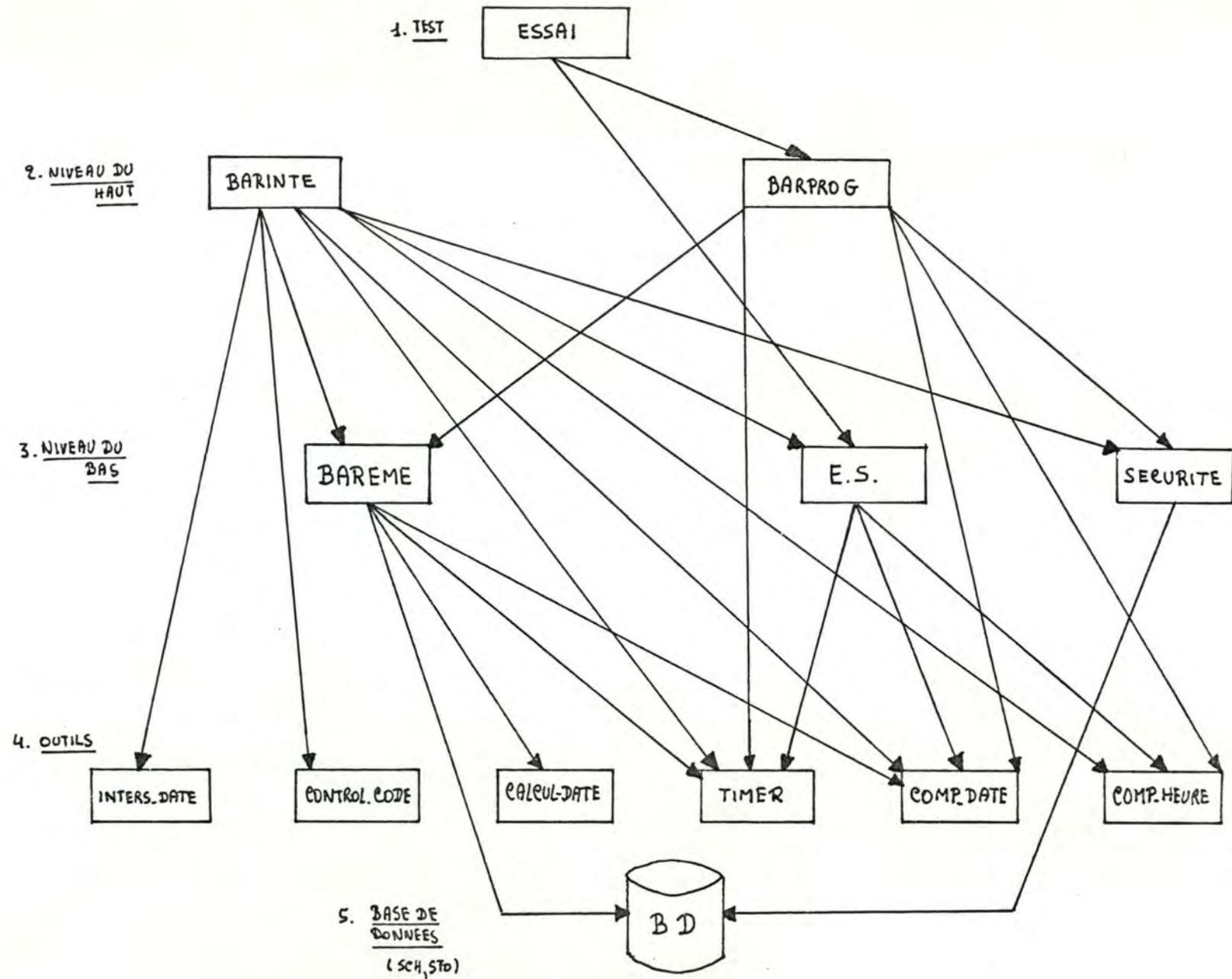
- TIMER.COB qui permet d'obtenir l'heure et la date du système;
- INTERS-DATE.COB qui permet d'obtenir l'intersection d'une série de dates;
- COMP-DATE.COB qui permet de comparer deux dates;
- COMP-HEURE.COB qui permet de comparer deux heures;
- CALCUL-DATE qui à partir d'une date et d'un nombre de jours calcule une deuxième date;
- CONTROL-CODE.COB qui effectue les contrôles de type et de longueur d'une valeur de code pour un barème. Comme vous l'avez remarqué, celui-ci ne concerne pas les dates et heures mais il est utile .

Il reste deux fichiers pour le niveau du haut. Ceux-ci se servent de ceux que l'on vient d'exposer. Nous avons donc:

- BARINTE.COB pour le module de gestion interactive de barèmes;
- BARPROG.COB pour le module de gestion de barèmes par programme.

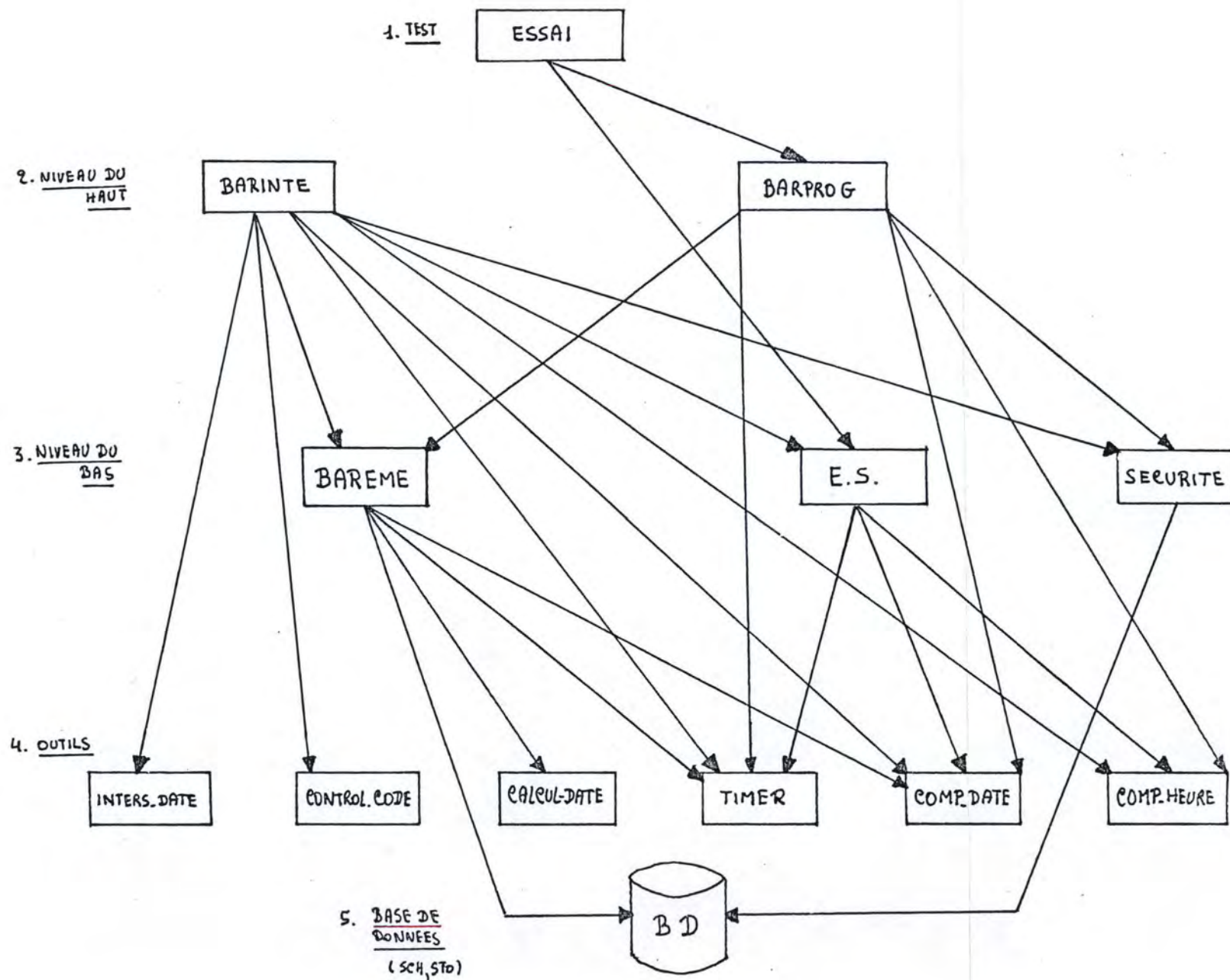
Vous pourrez consulter à la page suivante un graphe des appels entre programmes et ensuite l'ensemble de ces fichiers commentés.

Vous disposez aussi dans cette annexe du fichier ESSAI.COB qui permet de tester la gestion de barèmes par programmes.

Graph des appels.

1/ Test de la gestion par programmes

Graph des appels.



1/ Test de la gestion par programmes

Programme ESSAI .COB

IDENTIFICATION DIVISION.

PROGRAM-ID. TEST_GESTION_PAR_PROG.

AUTHOR. E EVRARD.

DATE-WRITTEN. 04/07/1986.

**
** TEST DE LA GESTION DE BAREMES PAR PROGRAMME **
**

** Ce programme a pour but de tester la gestion de baremes par programmes.
** Il s'agit donc d'interroger le module de gestion de baremes et de voir
** comment celui-ci reagit aux differentes demandes qu'on lui fait. Deux
** questions peuvent donc etre posees au module. Si pour une raison ou pour
** une autre, on ne peut satisfaire la demande, un numero d'erreur est renvoi
** au programme appelant.

**
** Question 1 Recherche selon critere.
** -----

** Arguments d'entree:

**
** NOM_UTIL : nom de l'utilisateur
** PSEUD_UTIL : mot de passe de l'utilisateur
** OPT_MENU : numero de la question posee (ici 1)
** LIBLU : libelle du bareme sur lequel on desire effectuer la recherche
** TAB_CRIT : tableau contenant le critere de recherche
** LCRIT : longueur de ce tableau
**

** Arguments de sortie:

**
** TABLEAU_VALS : tableau contenant les valeurs qui resultent de la recherche
** NLIG : longueur de ce tableau
** I : numero du tableau resultant
** CLE : identificateur dont la gestion de bareme a besoin pour continuer sa recherche
** TERM : indicateur qui nous dit oui(=1) ou non(=0) la recherche est terminee
** LIBBAR : libelle du bareme objet de la recherche
** TABLEAU_LIBS : tableau contenant les libelles des colonnes (c_a_d des baremes composants) du tableau des valeurs a sortir a l'ecran
** NB : longueur de ce tableau
** NUM_ERS : numero de l'erreur si l'on ne peut satisfaire la demande (il vaut 0 s'il peut le faire)
**

ENVIRONMENT DIVISION.

CONFIGURATION SECTION.

SOURCE-COMPUTER. vax-11.

OBJECT-COMPUTER. vax-11.

DATA DIVISION.

WORKING-STORAGE SECTION.

```

01 REP PIC X(3).
01 ID_UTIL PIC X(20).
01 HUH_MESS PIC 9(2).
01 OPE_MENU PIC 9(1).
01 ETAT1 PIC X(3).
01 ETAT2 PIC X(3).
01 ETAT3 PIC X(3).
01 TIME_FIELD PIC X(10).
01 HOUR_FIELD PIC X(5).
01 STOPPE PIC 9(1).
01 EXISTE PIC 9(1).
01 EXISTEC PIC 9(1).
01 EXCPT1 PIC 9(1).
01 ID_BAR PIC X(6).
01 HUH_BAR PIC X(6).
01 HLC PIC 9(2).
01 LIBC_BAR.
    02 ELEMENT OCCURS 1 TO 10 TIMES DEPENDING ON HLC.
        03 LIPELLEC PIC X(50).
        03 REC_LINCC PIC X(2).
01 DTC_CR PIC X(10).
01 TP_VAL PIC X(2).
01 TPC_VAL PIC X(2).
01 LG_VAL PIC X(2).
01 LGC_VAL PIC X(2).
01 LG_VAL_CAL PIC 9(2).
01 JB_VERS PIC X(2).
01 JBC_VERS PIC X(2).
01 DE_RET PIC X(4).
01 DRC_RET PIC X(4).
01 JBC PIC 9(2).
01 NBP PIC 9(2).
01 TAB_IDS.
    02 NBAR PIC X(6) OCCURS 99 TIMES.
01 TAB_IDC.
    02 NBARC PIC X(6) OCCURS 99 TIMES.
01 TAB_BARS.
    02 ELEMENT OCCURS 1 TO 10 TIMES DEPENDING ON NB.
        03 IDENT_BAR PIC X(6).
        03 POLE_BAR PIC X(50).
01 TAB_BARC.
    02 ELEMENT OCCURS 1 TO 10 TIMES DEPENDING ON NB.
        03 IDENT_BARC PIC X(6).
        03 POLE_BARC PIC X(50).
01 TAB_BARD.
    02 IDENT_BARD;PIC X(6);OCCURS 10 TIMES.

```

*

```

01 Z PIC 9(2).
01 I PIC 9(2).
01 J PIC 9(2).
01 HC PIC 9(2).
01 HCC PIC 9(2).
01 HVC PIC 9(2).
01 TAB_REL.
    02 ELEMENT OCCURS 10 TIMES.
        03 NREL PIC X(12).
        03 PREL PIC X(50).
01 ID_VALEUR.
    02 IDV_PART1 PIC X(6).
    02 IDV_PART2 PIC X(6).
01 ID_REL ; REDEFINES ID_VALEUR ; PIC X(12).
01 IDC_VALEUR.
    02 IDVC_PART1 PIC X(6).
    02 IDVC_PART2 PIC X(6).
01 HUH_REL ; REDEFINES IDC_VALEUR ; PIC X(12).
01 DTC_DB PIC X(10).
01 HRC_DB PIC X(5).
01 DTC_FH PIC X(10).
01 FINI PIC 9(1).
01 HUH_UTIL PIC X(10).
01 PSWD_UTIL PIC X(10).
01 HUH_ERR PIC 9(2).
01 VAL_CODE PIC X(6).
01 HL PIC 9(2).
01 LIBS_BAR.
    02 ELEMENT OCCURS 1 TO 10 TIMES DEPENDING ON HL.
        03 LIPELLE PIC X(50).
        03 REC_LING PIC X(2).
01 DT_CR PIC X(10).
01 HB PIC 9(2).
01 HUH_PROC PIC 9(2).
01 LIBLU PIC X(50).
01 IV PIC 9(2).
01 TAB_RELS.
    02 ELEMENT OCCURS 10 TIMES.
        03 NREL PIC X(12).
        03 PREL PIC X(50).
01 TAB_CRIT.
    02 ELEMENT OCCURS 10 TIMES.
        03 TCR_BAR PIC X(6).
        03 TCR_VAL PIC X(6).
01 LCRT PIC 9(2).
01 DT_DB PIC X(10).
01 HR_DB PIC X(5).
01 DT_FH PIC X(10).
01 TABLEAU_VALS.
    02 ELT1; OCCURS 08 TIMES.
        03 ELT2 PIC X(6); OCCURS 11 TIMES.
01 HLG PIC 9(2).
01 TABLEAU_LIBS.
    02 LIBS; PIC X(12); OCCURS 10 TIMES.
01 LIBBAR PIC X(50).
01 TERMI PIC 9(1).
01 W PIC 9(2).
01 CLE PIC X(12).
*
```



```
*****
PROCEDURE DIVISION.
*****
```

```
*****
*****
```

```
NIVEAU1 SECTION.
*-----
```

```
*****
*****
```

```
PROG_GEN.
```

```
** Voici ici le programme general qui affiche le menu et branche vers l'opti
** que l'utilisateur demande.
```

```
MOVE 01 TO NUM_PROC.PERFORM CALL_ES1.MOVE 00 TO TERM1.
DISPLAY"".DISPLAY"".DISPLAY"".DISPLAY"".
DISPLAY"Voulez-vous tester:".
DISPLAY"".DISPLAY"".DISPLAY"".
DISPLAY"      (1) La recherche selon critere ?".
DISPLAY"".DISPLAY"".DISPLAY"".
DISPLAY"      (2) L'obtention de(s) libelle(s) d'une valeur de code ?
DISPLAY"".DISPLAY"".DISPLAY"".
DISPLAY"      (3) Sortir du programme."
DISPLAY"".DISPLAY"".DISPLAY"".
DISPLAY"Votre reponse ? " WITH NO ADVANCING.
ACCEPT OPT_MENU.
IF OPT_MENU = 1 PERFORM RECHERCHE_CRITERE.
IF OPT_MENU = 2 PERFORM OBTENTION_LIBELLES.
IF OPT_MENU = 3 MOVE 01 TO NUM_PROC.PERFORM CALL_ES1 STOP RUN.
GO TO PROG_GEN.
```

```
*****
*
```


NIVEAU2 SECTION.

RECHERCHE-CRITERE.

** Cette primitive doit repondre a la question 1. Elle procede a la lecture
** des arguments d'entree appropriees, fait appel au module "gestion par pro-
** grammes" pour obtenir ses reponses.

```
MOVE 59 TO NUM_MESS.MOVE 00 TO NUM_PROC.PERFORM CALL_ES1.  
DISPLAY "Nom d'utilisateur : " WITH NO ADVANCING.  
ACCEPT NOM_UTIL.  
DISPLAY "".  
DISPLAY "Mot de passe      : " WITH NO ADVANCING.  
ACCEPT PSWD_UTIL.  
DISPLAY "".  
DISPLAY "Donnez le libelle du bareme : " WITH NO ADVANCING.  
ACCEPT LIBEL.  
DISPLAY "".  
MOVE 0 TO FINI.  
MOVE 59 TO NUM_MESS.MOVE 00 TO NUM_PROC.PERFORM CALL_ES1.  
PERFORM LECT_CRIT VARYING I FROM 1 BY 1 UNTIL FINI=1.  
PERFORM BOUCLE_TABLEAUX UNTIL TERMI=01.
```

*

OBTENTION-LIBELLES.

** Cette primitive doit repondre a la question 2. Elle procede a la lecture
** des arguments d'entree appropries, fait appel au module "gestion par pro-
** grammes" pour obtenir ses reponses, et imprime les resultats.
** Remarque: par facilite, on a reutilise ici le module "entrees/sorties"
** developpe pour la gestion interactive de baremes.

```
MOVE 63 TO NUM_MESS.MOVE 00 TO NUM_PROC.PERFORM CALL_ES1.
DISPLAY "Nom l'utilisateur : " WITH NO ADVANCING.
ACCEPT NOM_UTIL.
DISPLAY ".
DISPLAY "Mot de passe      : " WITH NO ADVANCING.
ACCEPT PSWD_UTIL.
DISPLAY ".
DISPLAY "Donnez le libelle du bareme : " WITH NO ADVANCING.
ACCEPT LIBLU.
DISPLAY ".
DISPLAY "Donnez la valeur de code : " WITH NO ADVANCING.
ACCEPT VAL_CODE.
MOVE 00 TO NUM_ERR.
CALL "GESTION-BAREME-PAR-PROGRAMME" USING BY REFERENCE
      OPT_MENU, NOM_UTIL, PSWD_UTIL, LIBLU, TAB_CRIT, LCRIT,
      TABLEAU_VALS, NLIG, TERMI, CLE, W, LIBBAR, TABLEAU_LIBS, U
      VAL_CODE, DT_CR, DT_DB, HR_DB, DT_FN, LIBS_BAR, NL, TAB_RE
      HV, NUM_ERR.
MOVE 63 TO NUM_MESS.MOVE 00 TO NUM_PROC.PERFORM CALL_ES1.
IF NUM_ERR = 00 MOVE 1 TO EXISTE
      MOVE 20 TO NUM_PROC.PERFORM CALL_ES1
ELSE DISPLAY "Erreur numero ", NUM_ERR PERFORM LEGENDE.
MOVE 13 TO NUM_PROC.PERFORM CALL_ES1.
```

NIVEAU3 SECTION.

BOUCLE-TABLEAUX.

** Ici se trouve l'appel au module "Gestion par programmes" pour la question
** ainsi que l'appel a l'impression des resultats. Le branchement vers cette
** primitive se fera tant que TERMI=00 c-a-d tant que la recherche n'est pas
** finie.

```
MOVE 00 TO NUM_ERR.
CALL "GESTION-BAREME-PAR-PROGRAMME" USING BY REFERENCE
      OPT_MENU, NOM_UTIL, PSWD_UTIL, LIBLU, TAB_CRIT, LCRIT,
      TABLEAU_VALS, NLIG, TERMI, CLE, W, LIBBAR, TABLEAU_LIBS, U
      VAL_CODE, DT_CR, DT_DB, HR_DB, DT_FN, LIBS_BAR, NL, TAB_RE
      HV, NUM_ERR.
```

*


```

MOVE 59 TO NUM_MESS. MOVE 00 TO NUM_PROC. PERFORM CALL_ES1.
IF NUM_ERR = 00
    PERFORM IMPRESSION_RESULT
ELSE DISPLAY " " MOVE 1 TO TERNI
    DISPLAY "Erreur numero ", NUM_ERR PERFORM LEGENDE
    MOVE 13 TO NUM_PROC
    PERFORM CALL_ES1.

```

IMPRESSION_RESULT.

** Impression d'un resultat vide ou du tableau resultant (parties gauche et
 ** droite si plus de 5 colonnes)

```

IF FLIG=0 AND M=1
    MOVE 69 TO NUM_MESS MOVE 00 TO NUM_PROC PERFORM CALL_ES1
    MOVE 13 TO NUM_PROC PERFORM CALL_ES1
ELSE MOVE "1" TO REP
    PERFORM CHOIX_TAB UNTIL REP="".

```

CHOIX_TAB.

** Sous bloc de la primitive precedante qui lit le desir de l'utilisateur
 ** de voir le tableau de gauche ou de droite s'il y a plus de 5 colonnes
 ** et qui sort le tableau a l'ecran a l'aide d'un appel au module "entrees/
 ** sorties" deja developpe.

```

IF REP="G"
    MOVE 24 TO NUM_PROC PERFORM CALL_ES1.
IF REP="D"
    MOVE 25 TO NUM_PROC PERFORM CALL_ES1.
MOVE 00 TO NUM_PROC.
IF REP="G" AND NB>5
    MOVE 53 TO NUM_MESS.
IF REP="G" AND NB<6
    MOVE 05 TO NUM_MESS.
IF REP="D"
    MOVE 35 TO NUM_MESS.
PERFORM CALL_ES1.
MOVE 21 TO NUM_PROC. PERFORM CALL_ES1.
IF REP NOT = "G" AND REP NOT = "D" AND REP NOT = ""
    MOVE "1" TO REP.
IF NB<5 AND REP = "D"
    MOVE "" TO REP.
IF NB<5 AND REP = "G"
    MOVE "" TO REP.

```

*

LECT_CRIT.

** Cette primitive doit lire une partie du critere de recherche et
** voir s'il y a encore une autre partie a lire.

```
DISPLAY "Donnez le numero du bareme composant ? " WITH NO ADVANCING.
ACCEPT TCR_BAR(I).
DISPLAY "Donnez la valeur de code composante ? " WITH NO ADVANCING.
ACCEPT TCR_VAL(I).
DISPLAY "Une autre entree ? " WITH NO ADVANCING.
ACCEPT REP.
IF REP="NON" OR REP="N" MOVE 1 TO FINI
                     MOVE I TO LCRIT
END-IF.
```

LEGENDE.

** Cette primitive sort a l'ecran la signification des numeros d'erreurs.

```
DISPLAY"".
DISPLAY"00 = Pas d'erreur.".
DISPLAY"01 = Utilisateur pas reconnu.".
DISPLAY"02 = Bareme inexistant.".
DISPLAY"03 = L'utilisateur n'a pas acces a ce bareme.".
DISPLAY"04 = Il n'y a aucune valeur dans ce bareme.".
DISPLAY"05 = Recherche interdite sur les bareme simples.".
DISPLAY"06 = Mauvais bareme composant.".
DISPLAY"07 = Valeur composante pas encore d'actualite.".
DISPLAY"08 = Une valeur composante n'est plus d'actualite.".
DISPLAY"09 = Une valeur composante est inexistante.".
DISPLAY"10 = Aucun resultat.".
DISPLAY"11 = La valeur de code n'existe pas.".
DISPLAY"12 = Il n'y a pas de bareme dans le systeme.".
DISPLAY"13 = Le numero de l'option demandee est incorrect.".
```

CALL_ES1.

** Ici se trouve l'appel au module ES que l'on utilise par facilite dans ce
** programme.

```
CALL "MODULE_ENTREE_SORTIE" USING BY REFERENCE
    NUM_PROC,NUM_MESS,ID_BAR,LIBS_BAR,NL,
    TAB_BARS,NB,NBP,DT_CR,TP_VAL,LG_VAL,
    NB_VERS,DR_RET,LIBDU,EXISTE,FINI,I,
    OPT_MENU,DT_DB,IR_DB,DT_EN,VAL_CODE,
    ID_REL,NV,TAB_RELS,REP,TABEAU_VALS,NLIG,
    TABLEAU_LIBS,LIBBAR,ID_UTIL.
```

2/ Niveau du haut .

Programme BARINTE.COB

IDENTIFICATION DIVISION.

PROGRAM-ID. GESTION-INTERACTIVE-BAREME.

AUTHOR. E EVRARD.

DATE-WRITTEN. 16/04/1986.

**
** GESTION INTERACTIVE DE BAREMES **
**

** Ce programme a pour but de repondre interactivement aux traitements
** qui ont ete developpes dans l'analyse fonctionnelle.

ENVIRONNEMENT DIVISION.

CONFIGURATION SECTION.

SOURCE-COMPUTER. vax_11.

OBJECT-COMPUTER. vax_11.

DATA DIVISION.

WORKING-STORAGE SECTION.

** Remarque: Pour les appels au module BAREME et ENTREES/SORTIES, vu la
** difficulte de travailler avec des variables globales COBOL
** et vu le type de memoire, il a ete juge bon de developper deux
** jeux de variables.

01 ID_UTIL PIC X(20).
01 A PIC 9(2).
01 CLC PIC X(12).
01 ETAT1 PIC X(3).
01 ETAT2 PIC X(3).
01 ETAT3 PIC X(3).
01 TIME_FIELD PIC X(10).
01 HOUR_FIELD PIC X(5).
01 DATE1 PIC X(10).
01 DATE2 PIC X(10).


```

01 HEURE PIC X(5).
01 TAB_DATES.
    02 ELEMENT OCCURS 1 TO 10 TIMES DEPENDING ON NV.
        03 DAT_DBT PIC X(10).
        03 HEU_DBT PIC X(5).
        03 DAT_FIN PIC X(10).
01 STOPPE PIC 9(1).
01 EC PIC 9(1).
01 NUH_NESS PIC 9(2).
01 EXISTE PIC 9(1).
01 EXISTEC PIC 9(1).
01 EXCEPT1 PIC 9(1).
01 EXCPT PIC 9(1).
01 ID_BAR PIC X(6).
01 NUH_BAP PIC X(6).
01 VAL_CODE PIC X(6).
01 NL PIC 9(2).
01 NLC PIC 9(2).
01 LIBS_BAP.
    02 ELEMENT OCCURS 1 TO 10 TIMES DEPENDING ON NL.
        03 LIBELLE PIC X(50).
        03 REC_LING PIC X(2).
01 LIBC_BAP.
    02 ELEMENT OCCURS 1 TO 10 TIMES DEPENDING ON NLC.
        03 LIBELLEC PIC X(50).
        03 REC_LINGC PIC X(2).
01 LO PIC 9(2).
01 TAB_LIBS.
    02 ELEMENT OCCURS 1 TO 20 TIMES DEPENDING ON LO.
        03 LIBBR PIC X(6).
        03 LIBOR PIC X(50).
01 DT_CR PIC X(10).
01 DFC_CR PIC X(10).
01 TP_VAL PIC X(2).
01 TPC_VAL PIC X(2).
01 LG_VAL PIC X(2).
01 LGC_VAL PIC X(2).
01 LG_VAL_CAL PIC 9(2).
01 NB_VERS PIC X(2).
01 NBC_VERS PIC X(2).
01 NB_VERS_CAL PIC 9(2).
01 DR_RET PIC X(1).
01 DRC_RET PIC X(4).
01 DR_RET_CAL PIC 9(4).
01 NB PIC 9(2).
01 NBC PIC 9(2).
01 NBP PIC 9(2).
01 TAB_BAPS.
    02 ELEMENT OCCURS 1 TO 10 TIMES DEPENDING ON NB.
        03 IDENT_FAP PIC X(6).
        03 POLE_BAR PIC X(50).
        03 DUP PIC 9(4).
01 TAB_BARC.
    02 ELEMENT OCCURS 1 TO 10 TIMES DEPENDING ON NB.
        03 IDENT_BARC PIC X(6).
        03 POLE_BARC PIC X(50).
        03 DURE PIC 9(4).
01 TAB_BAPP.
    02 IDENT_BAPP;PIC X(6);OCCURS 10 TIMES.
01 Z PIC 9(2).

```

```

01 I PIC 9(2).
01 J PIC 9(2).
01 REP PIC X(3).
01 HUM_PRODC PIC 9(2).
01 OPT_MENU PIC 9(1).
01 LIBLU PIC X(50).
01 HC PIC 9(2).
01 MCC PIC 9(2).
01 TAB_IDS.
    02 MBAR PIC X(6) OCCURS 99 TIMES.
01 TAB_IDC.
    02 MBARC PIC X(6) OCCURS 99 TIMES.
01 TAB_NB.
    02 TNR PIC 9(2) OCCURS 99 TIMES.
01 HV PIC 9(2).
01 HVC PIC 9(2).
01 TAB_RELS.
    02 ELEMENT OCCURS 10 TIMES.
    03 NREL PIC X(12).
    03 PREL PIC X(50).
01 TAB_RELC.
    02 ELEMENT OCCURS 10 TIMES.
    03 NRELC PIC X(12).
    03 PRELC PIC X(50).
01 ID_VALEUR.
    02 IDV_PART1 PIC X(6).
    02 IDV_PART2 PIC X(6).
01 ID_REL ; REDEFINES ID_VALEUR ; PIC X(12).
01 IDC_VALEUR.
    02 IDVC_PART1 PIC X(6).
    02 IDVC_PART2 PIC X(6).
01 HUM_REL ; REDEFINES IDC_VALEUR ; PIC X(12).
01 DT_DB PIC X(10).
01 DTC_DB PIC X(10).
01 HR_DB PIC X(5).
01 HRC_DB PIC X(5).
01 DE_FN PIC X(10).
01 DTC_FN PIC X(10).
01 FINI PIC 9(1).
01 TABLEAU_VALS.
    02 FLT1 OCCURS 08 TIMES.
    03 ELT2 PIC X(6) ; OCCURS 11 TIMES.
01 NLIG PIC 9(2).
01 TABLEAU_LIPS.
    02 LIPS ; PIC X(12) ; OCCURS 10 TIMES.
01 LIBBAR PIC X(50).
01 N PIC 9(2).
01 TERNI PIC 9(1).
01 W PIC 9(2).

```

```

*****
PROCEDURE DIVISION.
*****

```

```

*****
*****

```

```

NIVEAU1 SECTION.
*-----

```


PROG-IDENT.

** Cette primitive a pour but d'essayer d'identifier un utilisateur dans le
** systeme tant que celui-ci n'a pas ete formellement reconnu.

MOVE 2 TO STOPPE.

PERFORM IDENTIFIER UNTIL STOPPE=1.

GO TO PROG-TEST.

*

PROG-TEST.

** Cette primitive correspond au programme general du module. Il doit
** identifier l'option desirée par l'utilisateur et executer la proce-
** dure adéquate.

```
MOVE 01 TO NUM_PROC.PERFORM CALL-ES1.  
MOVE 00 TO NUM_PROC.  
MOVE 01 TO NUM_MESS.PERFORM CALL-ES1.  
MOVE 15 TO NUM_PROC.PERFORM CALL-ES1.  
IF OPT_MENU > 1 AND OPT_MENU < 10  
  MOVE 02 TO NUM_PROC PERFORM CALL-MBAR1  
  IF EXISTE = 0  
    MOVE 01 TO NUM_PROC  
    PERFORM CALL-ES1  
    MOVE 47 TO NUM_MESS  
    MOVE 00 TO NUM_PROC  
    PERFORM CALL-ES1  
    MOVE 13 TO NUM_PROC  
    PERFORM CALL-ES1  
    GO TO PROG-TEST
```

*Si on regarde s'il existe
des barèmes dans le système
pour les hautements suivant:
n° 2 → n° 3.*

```
END-IF  
END-IF.  
IF OPT_MENU = 0 MOVE 01 TO NUM_PROC  
  PERFORM CALL-ES1  
  STOP RUN.  
IF OPT_MENU = 1 PERFORM CREER_BAREME.  
IF OPT_MENU = 2 MOVE 01 TO NUM_PROC PERFORM CALL-ES1  
  MOVE 0 TO FINI  
  MOVE 62 TO NUM_MESS MOVE 00 TO NUM_PROC  
  PERFORM CALL-ES1  
  PERFORM CONTROL_EXIST_BAR UNTIL FINI=1  
  MOVE NUM_BAR TO ID_BAR  
  MOVE 02 TO NUM_PROC PERFORM CALL-MBAR1  
  IF NB>0 MOVE 0 TO STOPPE  
    PERFORM CONTROL_VIDE VARYING J  
    FROM 1 BY 1 UNTIL J>NB  
    IF STOPPE = 1 MOVE 00 TO NUM_PROC  
      MOVE 34 TO NUM_MESS  
      PERFORM CALL-ES1  
      MOVE 13 TO NUM_PROC  
      PERFORM CALL-ES1  
      GO TO PROG-TEST
```

*identification
sur barème vide*

*Le barème est
composé mais
refusé car un de
barèmes compo-
ne contient pas
de valeur!*

```
END-IF  
END-IF  
MOVE 0 TO STOPPE  
PERFORM CREER_VALEURS UNTIL STOPPE=1.  
IF OPT_MENU = 3 PERFORM OBTENIR_LIBELLES_VALEUR.  
IF OPT_MENU = 4 PERFORM AFFICHER_BAREME.  
IF OPT_MENU = 5 PERFORM OBTENIR_COMPOSITION.  
IF OPT_MENU = 6 PERFORM OBTENIR_TOUS_NOMS.  
IF OPT_MENU = 7 PERFORM RECHERCHE_CRITERE.  
IF OPT_MENU = 8 PERFORM OBTENIR_ACCES.  
IF OPT_MENU = 9 PERFORM NETTOYAGE_SYSTEME.  
GO TO PROG-TEST.
```

*

NIVEAU2 SECTION.

IDENTIFIER.

** Cette primitive a pour but d'identifier, c-a-d d'accepter ou non q'un
 ** utilisateur se serve du systeme. L'identification se fait par la lecture
 ** d'un nom et d'un mot de passe a l'ecran.

```
MOVE SPACES TO ID_UTIL.
MOVE 26 TO NUM_PROC. PERFORM CALL_ES1.
MOVE 1 TO NUM_PROC. PERFORM CALL_SEC.
IF EXISTE = 1 MOVE 1 TO STOPPE MOVE 71 TO NUM_MESS
ELSE MOVE 70 TO NUM_MESS.
MOVE 00 TO NUM_PROC. PERFORM CALL_ES1.
MOVE 13 TO NUM_PROC. PERFORM CALL_ES1.
```

NETTOYAGE-SYSTEME.

** Cette primitive procede au nettoyage du systeme. Celui-ci comporte la
 ** verification de toutes les valeurs de code du systeme et l'elimination
 ** de toutes celles qui sont perimees. Comme precise dans l'analyse, on
 ** commencera par la revision des baremes composes, et ensuite, les simples.
 ** Une fois le nettoyage les valeurs perimees effectue, on procedera a la
 ** suppression des baremes n'ayant plus aucune valeur de code.

```
MOVE 01 TO NUM_PROC. PERFORM CALL_ES1.
MOVE 00 TO NUM_PROC. MOVE 81 TO NUM_MESS. PERFORM CALL_ES1.
MOVE 04 TO NUM_PROC. PERFORM CALL_MBAR1.
PERFORM EXTR_NB VARYING I FROM 1 BY 1 UNTIL I>NC.
PERFORM SUPP_VAL_COMP VARYING I FROM 1 BY 1 UNTIL I>NC.
PERFORM SUPP_VAL_SIMP VARYING I FROM 1 BY 1 UNTIL I>NC.
MOVE 0 TO J.
PERFORM SUPP_BAR VARYING I FROM 1 BY 1 UNTIL I>NC.
MOVE 0 TO LO.
IF LO>0
  MOVE 82 TO NUM_MESS MOVE 00 TO NUM_PROC
  PERFORM CALL_ES1
ELSE MOVE 83 TO NUM_MESS MOVE 00 TO NUM_PROC
  PERFORM CALL_ES1.
PERFORM AFF_SUPP VARYING I FROM 1 BY 1 UNTIL I>LO.
MOVE 13 TO NUM_PROC. PERFORM CALL_ES1.
```

AFF_SUPP.

** Cette primitive est une sous procedure de "NETTOYAGE-SYSTEME" et sert
 ** a afficher a l'ecran, l'identificateur et le libelle du bareme numero I
 ** (dans le tableau) que l'on vient de supprimer.

```
DISPLAY HI0BR(I), " ", LIBBR(I).
DISPLAY "".
```

 *

OBTENIR_ACCES.

** Cette primitive nous donne tous les libelles de baremes auxquels l'utilisateur
** a acces dans le systeme.

```
MOVE 04 TO NUM_PROC.PERFORM CALL_SEC.  
IF NC>0  
  MOVE 02 TO N  
  MOVE 0 TO FINI  MOVE 0 TO Z  
  PERFORM QUATRE_NUM_BAR UNTIL FINI=1  
  PERFORM EFFACE_TAB_IDS VARYING I FROM 1 BY 1 UNTIL I>NC  
ELSE MOVE 75 TO NUM_MESS MOVE 00 TO NUM_PROC PERFORM CALL_ES1  
  MOVE 78 TO NUM_MESS MOVE 00 TO NUM_PROC PERFORM CALL_ES1  
  MOVE 13 TO NUM_PROC PERFORM CALL_ES1.
```

RECHERCHE_CRITERE.

** Cette primitive realise la recherche selon critere comme identifie dans
** l'analyse fonctionnelle. Le but est d'obtenir un ou plusieurs tableau(x)
** de valeurs de code resultant d'une recherche sur base d'un critere lu
** au depart. Cette recherche peut faire l'objet de refus (ex: interdite
** sur les baremes simples) ou peut donner un resultat vide.

```
PERFORM INIT_TABL_LIBS VARYING I FROM 1 BY 1 UNTIL I>10.  
PERFORM INIT_TABL_VALS VARYING I FROM 1 BY 1 UNTIL I>8.  
MOVE 00 TO NUM_PROC.MOVE 59 TO NUM_MESS.PERFORM CALL_ES1.  
MOVE 0 TO FINI.  
PERFORM CONTROL_EXIST_BAR UNTIL FINI=1.  
MOVE 10 TO NUM_PROC.MOVE NUM_BAR TO ID_BAR.  
PERFORM CALL_NBAR1.  
IF EXISTE = 0  MOVE 00 TO NUM_PROC  
               MOVE 59 TO NUM_MESS  
               PERFORM CALL_ES1  
               MOVE 48 TO NUM_MESS  
               MOVE 00 TO NUM_PROC  
               PERFORM CALL_ES1  
               MOVE 13 TO NUM_PROC  
               PERFORM CALL_ES1  
               GO TO PROG_TEST.  
MOVE 00 TO NUM_PROC.MOVE 59 TO NUM_MESS.PERFORM CALL_ES1.  
MOVE 02 TO NUM_PROC.PERFORM CALL_NBAR1.  
IF NB>0:  
  MOVE 0 TO NDP  MOVE 0 TO Z  
  PERFORM CONTROL_BAR_CRIT VARYING J FROM 1 BY 1 UNTIL J>NB  
  MOVE NDP TO NV  
  PERFORM CONTROL_REL_CRIT VARYING J FROM 1 BY 1 UNTIL J>NV  
  MOVE 0 TO TERMI  MOVE 1 TO W  
  MOVE LIBLU TO LIBBAR  
  PERFORM IMPRESSION_RESULT UNTIL TERMI=1  
ELSE MOVE 68 TO NUM_MESS MOVE 00 TO NUM_PROC PERFORM CALL_ES1  
  MOVE 13 TO NUM_PROC PERFORM CALL_ES1.
```

*

OBTENIR_LIBELLES_VALEUR.

** Le but de cette primitive est d'obtenir a l'ecran a partir d'une valeur de
** code lue et du nom du bareme auquel cette valeur appartient, les renseigne-
** ments concernant cette valeur (c-a-l ses dates et ses libelles existants)

```
MOVE 00 TO NUM_PROC.MOVE 63 TO NUM_MESS.PERFORM CALL_ES1.
MOVE 0 TO FINI.
PERFORM CONTROL_EXIST_BAR UNTIL FINI=1.
MOVE 10 TO NUM_PROC.MOVE NUM_BAR TO ID_BAR.
PERFORM CALL_MBAR1.
IF EXISTE = 0 MOVE 00 TO NUM_PROC
              MOVE 63 TO NUM_MESS
              PERFORM CALL_ES1
              MOVE 48 TO NUM_MESS
              MOVE 00 TO NUM_PROC
              PERFORM CALL_ES1
              MOVE 13 TO NUM_PROC
              PERFORM CALL_ES1
              GO TO PROG_TEST.
MOVE 00 TO NUM_PROC.MOVE 63 TO NUM_MESS.PERFORM CALL_ES1.
MOVE 00 TO NUM_PROC.MOVE 43 TO NUM_MESS.PERFORM CALL_ES1.
MOVE 13 TO NUM_PROC.PERFORM CALL_ES1.
MOVE VAL_CODE TO IDV_PART2.
MOVE NUM_BAR TO IDV_PART1.
MOVE NUM_BAR TO ID_BAR.MOVE 02 TO NUM_PROC.
PERFORM CALL_MBAR1.
MOVE 00 TO NUM_PROC.MOVE 63 TO NUM_MESS.PERFORM CALL_ES1.
MOVE 08 TO NUM_PROC.PERFORM CALL_MBAR1.
IF EXISTE=1 MOVE 20 TO NUM_PROC
            PERFORM CALL_ES1
            MOVE 13 TO NUM_PROC
            PERFORM CALL_ES1
            MOVE HV TO H
            IF H>0
                PERFORM COPIE_REL_RELC VARYING J FROM 1 BY 1
                UNTIL J>N
                PERFORM SORTIR_REL_COMP VARYING Z FROM 1 BY 1
                UNTIL Z>N
            END-IF
ELSE MOVE 40 TO NUM_MESS
      MOVE 00 TO NUM_PROC
      PERFORM CALL_ES1
      MOVE 13 TO NUM_PROC
      PERFORM CALL_ES1.
```

*

CREER_BAREME.

** Le but de cette primitive est de creer a l'ecran un en-tete de bareme.
** Des verifications sont effectuees sur les differentes entrees. Les lectures
** d'un nouvel identificateur ou des libelles du bareme seront refusees et
** re_litrees ssi ils existent deja dans le systeme.

```
MOVE 00 TO HUM_PROC.MOVE 61 TO HUM_MESS.PERFORM CALL_ES1.
MOVE 0 TO FINI.
PERFORM CONTROL_ID UNTIL FINI=1.
MOVE 00 TO HUM_PROC.MOVE 61 TO HUM_MESS.PERFORM CALL_ES1.
MOVE 05 TO HUM_PROC.PERFORM CALL_ES1.
PERFORM LECT_LIS VARYING I FROM 1 BY 1 UNTIL I>NL.
MOVE 0 TO FINI.
MOVE 03 TO HUM_PROC.
PERFORM CALL_ES1 UNTIL FINI=1.
IF NBP>0 THEN PERFORM CONTROL_BAR_COMP VARYING I FROM 1 BY 1
UNTIL I>NBP.
MOVE NBP TO NB.
IF NB=0
    MOVE 28 TO HUM_PROC
    MOVE 0 TO FINI
    PERFORM CALL_ES1 UNTIL FINI=1
END-IF.
IF NB>0
    MOVE 9999 TO DR_RET
    PERFORM GIVE_DR_RET VARYING I FROM 1 BY 1 UNTIL I>NB
    MOVE 00 TO HUM_PROC
    MOVE 13 TO HUM_MESS
    PERFORM CALL_ES1
    DISPLAY DR_RET
END-IF.
MOVE 01 TO HUM_PROC.
PERFORM CALL_MEAR1.
MOVE 03 TO HUM_PROC.
PERFORM CALL_SEC.
MOVE 06 TO HUM_PROC.
MOVE 03 TO HUM_MESS.
PERFORM CALL_US1.
MOVE 13 TO HUM_PROC.PERFORM CALL_ES1.
```

*

AFFICHER_BAREME.

** Cette primitive a pour but, a partir du libelle d'un bareme, d'afficher ce
** bareme a l'ecran. L'affichage comporte non seulement le tableau des valeurs
** de code mais aussi les renseignements generaux concernant le bareme.

```
PERFORM INIT_TABL_LIBS VARYING I FROM 1 BY 1 UNTIL I>10.
PERFORM INIT_TABL_VALS VARYING I FROM 1 BY 1 UNTIL I>8.
MOVE 00 TO NUM_PROC. MOVE 64 TO NUM_MESS. PERFORM CALL_ES1.
MOVE 0 TO FINI.
PERFORM CONTROL_EXIST_BAR UNTIL FINI=1.
MOVE 00 TO NUM_PROC. MOVE 64 TO NUM_MESS. PERFORM CALL_ES1.
MOVE NUM_BAR TO ID_BAR.
MOVE 02 TO NUM_PROC.
PERFORM CALL_HBAR1.
MOVE LIBELLE(1) TO LIBBAR.
PERFORM COPIE_LIB_TABLEAU VARYING Z FROM 1 BY 1 UNTIL Z>NB.
MOVE 14 TO NUM_PROC. PERFORM CALL_ES1.
MOVE 00 TO NUM_PROC.
MOVE 05 TO NUM_MESS. PERFORM CALL_ES2.
MOVE 13 TO NUM_PROC. PERFORM CALL_ES2.
IF NB>0 MOVE 27 TO NUM_PROC PERFORM CALL_ES1.
IF NB>0 AND REP="0"
    MOVE 00 TO NUM_PROC MOVE 64 TO NUM_MESS PERFORM CALL_ES1
    MOVE NB TO N
    PERFORM SORTIR_BAR_COMP VARYING Z FROM 1 BY 1 UNTIL Z>N.
MOVE 10 TO NUM_PROC.
PERFORM CALL_HBAR1.
IF EXISTE = 0 MOVE 00 TO NUM_PROC MOVE 64 TO NUM_MESS
    PERFORM CALL_ES1 MOVE 48 TO NUM_MESS
    MOVE 00 TO NUM_PROC PERFORM CALL_ES1
    MOVE 13 TO NUM_PROC PERFORM CALL_ES1
ELSE MOVE 0 TO TERMI MOVE 1 TO W
    PERFORM IMPRESSION_BAREME UNTIL TERMI=1.
```

*

OBTENIR_COMPOSITION.

** Cette primitive a pour but d'obtenir la composition d'un bareme a partir
** d'un libelle de ce bareme. Si le bareme est simple, on obtiendra les
** libelles de tous les baremes qui l'admettent comme composant. Si le bareme
** est compose, on obtiendra les libelles des baremes qui composent ce bareme
** Remarque: On accedera d'abord aux identificateurs des baremes qui inter-
** viennent dans la composition, et a base de ces identificateurs, on obtiend
** ces libelles.

```
MOVE 00 TO NUM_PROC.MOVE 65 TO NUM_MESS.PERFORM CALL_ES1.
MOVE 0 TO FINI.
PERFORM CONTROL_EXIST_BAR UNTIL FINI=1.
MOVE 00 TO NUM_PROC.MOVE 65 TO NUM_MESS.PERFORM CALL_ES1.
MOVE NUM_BAR TO ID_BAR.
MOVE 02 TO NUM_PROC.
PERFORM CALL_NBARI.
MOVE 00 TO NUM_PROC.
IF NB = 0 MOVE 27 TO NUM_MESS
ELSE MOVE 28 TO NUM_MESS.
PERFORM CALL_ES1.
MOVE 11 TO NUM_PROC.
PERFORM CALL_ES1 VARYING I FROM 1 BY 1 UNTIL I>NL.
IF NB>0 MOVE 00 TO NUM_PROC
MOVE 22 TO NUM_MESS
PERFORM CALL_ES1
PERFORM EXTRAIRE_COMPOSANT VARYING J FROM 1 BY 1
UNTIL J>NB.
IF NB=0 MOVE 06 TO NUM_PROC
PERFORM CALL_NBARI
IF NC > 0
MOVE 00 TO NUM_PROC
MOVE 30 TO NUM_MESS
PERFORM CALL_ES1
PERFORM EXTRAIRE_COMPOSE VARYING J FROM 1 BY 1
UNTIL J>NC
ELSE MOVE 31 TO NUM_MESS
MOVE 00 TO NUM_PROC
PERFORM CALL_ES1.
MOVE 13 TO NUM_PROC.PERFORM CALL_ES1.
```

OBTENIR_TOUS_NOMS.

** Cette primitive a pour but d'obtenir a l'ecran tous les libelles de tous les
** baremes existant dans le systeme.

```
MOVE 04 TO NUM_PROC.MOVE 01 TO N.
PERFORM CALL_NBARI.MOVE 0 TO FINI.MOVE 0 TO Z.
PERFORM QUATRE_NUM_BAR UNTIL FINI=1.
PERFORM EFFACE_TAB_IDS VARYING I FROM 1 BY 1 UNTIL I>NC.
```

*

CREER_VALFURS.

** Cette primitive a pour but de creer, dans un bareme dont le libelle est lu
** a l'ecran, une valeur de code et ses renseignements generaux.

```
MOVE 00 TO NUM_PROC.MOVE 62 TO NUM_MESS.PERFORM CALL_ES1.
MOVE 0 TO FINI.
MOVE ID_BAR TO NUM_BAR.
PERFORM CONTROL_VAL UNTIL FINI=1.
MOVE ID_BAR TO IDV_PART1.
MOVE VAL_CODE TO IDV_PART2.
MOVE 00 TO NUM_PROC.MOVE 62 TO NUM_MESS.PERFORM CALL_ES1.
MOVE 17 TO NUM_PROC.PERFORM CALL_ES1.
IF NB>0 THEN PERFORM CONTROL_REL_COMP VARYING J FROM 1 BY 1
                                UNTIL J>NB.

IF NB = 0
    MOVE 00 TO NUM_PROC.MOVE 62 TO NUM_MESS.PERFORM CALL_ES1
    MOVE 22 TO NUM_PROC.PERFORM CALL_ES1.
```

MOVE NB TO NV.

```
IF NV>0
    CALL"INTERG_DATE"USING BY REFERENCE TAB_DATES,NV,DATE1,HEURE,DATE2
    CALL"TIMER"USING BY REFERENCE TIME_FIELD,HEURE
    MOVE DT_CR TO DT_DB
    MOVE DATE2 TO DT_FH
    MOVE HEURE TO HR_DB
    MOVE 23 TO NUM_PROC.PERFORM CALL_ES1
END-IF.
```

```
MOVE 07 TO NUM_PROC.
PERFORM CALL_MBARI.
MOVE 00 TO NUM_PROC.
MOVE 02 TO NUM_MESS.
PERFORM CALL_ES1.
MOVE 04 TO NUM_MESS.MOVE 00 TO NUM_PROC.PERFORM CALL_ES1.
MOVE 21 TO NUM_PROC.PERFORM CALL_ES1.
IF REP="JH" OR REP="I" MOVE 1 TO STOPPE
ELSE MOVE 0 TO STOPPE.
```

NIVEAU3 SECTION.

INIT_TABB_LIBS.

** Cette primitive a pour but d'initialiser le tableau TAB_LIBS destine a
** contenir une liste de libelles de baremes (+ identificateurs associes).

MOVE SPACES TO LIBS(I).

*

INIT_TABL_VALS.

** Cette primitive s'occupe d'initialiser une ligne du tableau des valeurs de
** code. Rappel: Ce tableau est destine a contenir le resultat d'une recherche
** selon critere.

PERFORM INIT_LIGNE VARYING J FROM 1 BY 1 UNTIL J>11.

INIT_LIGNE.

** Cette primitive doit initialiser un element du tableau decrit ci-dessus.

MOVE SPACES TO ELT2(I,J).

SUPP_BAR.

** Cette primitive verifie dans le systeme si le bareme, dont l'identificateur
** se trouve memorise dans NBAR(I), ne doit pas etre supprime du systeme. Il
** devra l'etre s'il ne contient plus aucune valeur.

MOVE NBAR(I) TO ID_BAR.
MOVE 10 TO NUM_PROC. PERFORM CALL_MBARI.
IF EXISTE=0
MOVE 02 TO NUM_PROC PERFORM CALL_MBARI
ADD 1 TO J
MOVE ID_BAR TO MIDBR(J)
MOVE LIBELLE(1) TO LIBBR(J)
MOVE 11 TO NUM_PROC
PERFORM CALL_MBARI.

EXTR_NB.

** Cette primitive doit extraire du systeme le nombre de baremes composants,
** bareme dont l'identifiant se trouve memorise dans NBAR(I) et mettre ce
** nombre dans TNB(I).

MOVE 02 TO NUM_PROC. MOVE NBAR(I) TO ID_BAR.
PERFORM CALL_MBARI. MOVE NB TO TNB(I).

SUPP_VAL_COMP.

** Cette primitive doit passer en revue toutes les valeurs de code du bareme
** compose dont l'identificateur se trouve memorise dans NBAR(I) et supprimer
** toutes celles qui sont perimees.

IF TNB(I) > 0
DISPLAY "Je passe en revue le bareme compose Nr ",NBAR(I)," !"
MOVE 13 TO NUM_PROC
MOVE NBAR(I) TO ID_BAR
PERFORM CALL_MBARI.

*

SUPP_VAL_SIMP.

** Cette primitive doit passer en revue toutes les valeurs de code du bareme
** simple dont l'identificateur se trouve memorise dans NBAR(I) et supprimer
** toutes celles qui sont perimees.

```
IF TMR(I) = 0
  DISPLAY "Je passe en revue le bareme simple Nr ",NBAR(I)," !"
  MOVE 13 TO NUN_PROC
  MOVE NBAR(I) TO ID_BAR
  PERFORM CALL_NBAR1.
```

GIVE_DR_RET.

** Cette primitive doit mettre DUR(I) dans DR_RET ssi DUR(I) est plus
** petit que DR_RET. A la fin DR_RET doit contenir la plus petite valeur
** du tableau des DUR(I). Ceci afin de realiser la contrainte sur les durees
** de retention. (Corp de boucle)

```
IF DUR(I) < DR_RET MOVE DUR(I) TO DR_RET.
```

DECT_LIB.

** Cette primitive doit lire un libelle, verifier s'il existe deja dans le
** systeme auquel cas il devra se voir refuse.
** Rappel: Le libelle d'un bareme doit etre identifiant.

```
MOVE 16 TO NUN_PROC.
PERFORM CALL_ES1.
MOVE LIBELLE(I) TO LIBLI.
MOVE 05 TO NUN_PROC.
PERFORM CALL_NBAR2.
IF EXISTED=1
  SUBTRACT 1 FROM I
  MOVE 34 TO NUN_LESS
  MOVE 00 TO NUN_PROC
  PERFORM CALL_ES1.
```

IMPRESSION_BAREME.

** Cette primitive accede au tableau suivant le valeurs du bareme dont
** l'identifiant se trouve dans ID_BAR et sort ce tableau a l'ecran.
** Si c'est un bareme compose dont le nombre de composants est superieur
** a 5, le tableau se divise en deux ecrans visibles selon le bon vouloir
** de l'utilisateur. (Commande "G" pour le tableau de gauche et "D" pour
** le tableau de droite).

```
MOVE 11 TO NUN_PROC. PERFORM CALL_NBAR1.
MOVE "G" TO REP.
IF PLIG > 0 PERFORM CHOIX_TAB UNTIL REP = ".
COMPUTE N = N + 1.
```

*

CHOIX_TAB.

** Cette primitive identifie selon la valeur de l'argument REP s'il faut sortir
** le tableau de gauche ou de droite a l'ecran, sort cet ecran et procede a un
** relecture de cette variable REP pour la fois suivante.

```
IF REP="G"
  MOVE 21 TO HUI_PROC PERFORM CALL_ES1.
IF REP="D"
  MOVE 25 TO HUI_PROC PERFORM CALL_ES1.
MOVE 00 TO HUI_PROC.
IF REP="G" AND NB>5
  MOVE 58 TO HUI_MESS.
IF REP="G" AND NB<6
  MOVE 05 TO HUI_MESS.
IF REP="D"
  MOVE 35 TO HUI_MESS.
PERFORM CALL_ES1.
MOVE 21 TO HUI_PROC. PERFORM CALL_ES1.
IF REP NOT = "G" AND REP NOT = "D" AND REP NOT = ""
  MOVE "G" TO REP.
IF NB<5 AND REP = "D"
  MOVE "" TO REP.
IF NB<5 AND REP = "G"
  MOVE "" TO REP.
```

CONTROL_REL_CRIT.

** Cette primitive a pour but de proceder a la lecture d'une valeur de code
** existante dans le bareme dont l'identifiant se trouve dans IDENT_BAP(J).
** Cette valeur de code constitue une partie du critere de la recherche.

```
MOVE 0 TO FINI. MOVE 02 TO N. MOVE 00 TO Z.
PERFORM VALID_VAL UNTIL FINI = 1.
MOVE HUI_REL TO HREL(J).
```

IMPRESSION_RESULT.

** Cette primitive accede au tableau suivant de valeurs concernant une recherche
** selon critere, et sort ce tableau a l'ecran de la meme maniere que
** l'affichage d'un bareme.

```
MOVE 12 TO HUI_PROC.
PERFORM CALL_HBAR1.
IF HUIG=0 AND N=1
  MOVE 69 TO HUI_MESS MOVE 00 TO HUI_PROC PERFORM CALL_ES1
  MOVE 13 TO HUI_PROC PERFORM CALL_ES1
ELSE MOVE "G" TO REP
  PERFORM CHOIX_TAB UNTIL REP=""
  COMPUTE N = N + 1.
```

*

CONTROL_BAR_CRIT.

** Cette primitive a pour but d'identifier les baremes qui font partie de
** la recherche selon critere.

```
MOVE 59 TO NUM_MESS. MOVE 00 TO NUM_PROC. PERFORM CALL_ES1.  
MOVE IDENT_BAR(J) TO NUM_BAR. MOVE 02 TO NUM_PROC.  
PERFORM CALL_IBAR2. MOVE 11 TO NUM_PROC.  
PERFORM CALL_ES2 VARYING I FROM 1 BY 1 UNTIL I>NLC.  
MOVE LIBELLOC(1) TO LIBS(J).  
MOVE "" TO REP.  
PERFORM LECT_REP UNTIL REP="0" OR REP="N".  
IF REP="0" COMPUTE NBP = NBP + 1  
      MOVE IDENT_BAR(J) TO IDENT_BARP(NBP).  
IF J=NB AND NBP=0 MOVE 60 TO NUM_MESS. MOVE 00 TO NUM_PROC.  
      PERFORM CALL_ES1. MOVE 13 TO NUM_PROC.  
      PERFORM CALL_ES1.  
      GO TO PROG_TEST.
```

LECT_REP.

** Cette primitive sort un message et procede a la lecture de REP.
** Son existence se justifie par la presence d'une boucle au niveau
** superieur.

```
MOVE 19 TO NUM_MESS. MOVE 00 TO NUM_PROC. PERFORM CALL_ES1.  
MOVE 21 TO NUM_PROC. PERFORM CALL_ES1.
```

CONTROL_VIDP.

** Cette primitive sert a controler s'il existe au moins une valeur dans le
** bareme dont l'identificateur se trouve dans IDENT_BAR(J).

```
MOVE IDENT_BAR(J) TO NUM_BAR.  
MOVE 10 TO NUM_PROC.  
PERFORM CALL_IBAR2.  
IF EXISTED=0 MOVE 1 TO STOPPE.
```

*

QUATRE_HON_BAR.

** Cette primitive a pour but de sortir a l'ecran les noms (libelles) des
** baremes dont les identificateurs se trouvent dans le tableau TAB_IDS.
** Elle le fait par ecran de quatre baremes maximum.

```
MOVE 01 TO HUM_PROC.PERFORM CALL_ES1.  
IF N = 01 MOVE 32 TO HUM_NESS.  
IF N = 02 MOVE 75 TO HUM_NESS.  
MOVE 00 TO HUM_PROC.  
PERFORM CALL_ES1.  
MOVE 0 TO EC.  
PERFORM SORTIR_HON_BAR VARYING J FROM 1 BY 1 UNTIL EC=1.  
MOVE 86 TO HUM_NESS.MOVE 00 TO HUM_PROC.PERFORM CALL_ES1.  
MOVE 21 TO HUM_PROC.PERFORM CALL_ES1.  
IF REP = "S" GO TO PROG_TEST.
```

EXTRAIRE_COMPOSE.

** Cette primitive a pour but de sortir a l'ecran les libelles du bareme
** compose dont l'identificateur se trouve dans NBAR(I). Cette primitive
** fait partie de l'option "OBTENIR_COMPOSITION" et les identificateurs
** en question sont ceux des baremes qui admettent un bareme simple (de-
** termine au niveau precedent) comme composant.

```
MOVE NBAR(J) TO HUM_BAR.  
MOVE 02 TO HUM_PROC.  
PERFORM CALL_HBAR2.  
MOVE NLC TO IL.  
PERFORM COPIE_LIBC_LIBS VARYING Z FROM 1 BY 1 UNTIL Z>NL.  
MOVE 11 TO HUM_PROC.  
PERFORM CALL_ES1 VARYING I FROM 1 BY 1 UNTIL I>NL.  
DISPLAY "".
```

EXTRAIRE_COMPOSANT.

** Cette primitive a pour but de sortir a l'ecran les libelles du bareme simple
** dont l'identificateur se trouve dans IDENT_BAR(J). Les identificateurs en
** question sont ceux des baremes simples qui composent un bareme determine
** au niveau precedent.

```
MOVE IDENT_BAR(J) TO HUM_BAR.  
MOVE 02 TO HUM_PROC.  
PERFORM CALL_HBAR2.  
MOVE NLC TO NL.  
PERFORM COPIE_LIBC_LIBS VARYING Z FROM 1 BY 1 UNTIL Z>NL.  
MOVE 11 TO HUM_PROC.  
PERFORM CALL_ES1 VARYING I FROM 1 BY 1 UNTIL I>NL.  
DISPLAY "".
```

*

CONTROL_ID.

** Cette primitive a pour but de verifier s'il existe dans le systeme un bareme
** dont l'identificateur est contenu dans ID_BAR. Si cet identificateur existe
** deja, un message d'erreur apparaitra a l'ecran car cet identifiant doit etre
** refuse.

```
MOVE 02 TO NUM_PROC.PERFORM CALL_ES1.  
MOVE 03 TO NUM_PROC.  
PERFORM CALL_MBAR1.  
IF EXISTE = 1 MOVE 15 TO NUM_HESS  
MOVE 00 TO NUM_PROC  
PERFORM CALL_ES1  
ELSE MOVE 1 TO FINI.
```

CONTROL_BAR_CCHP.

** S'il existe des baremes dans le systeme, cette primitive a pour but lors de
** la creation d'un bareme compose, de determiner un bareme composant.
** Ce bareme composant doit exister dans le systeme, doit etre simple et ne
** doit pas avoir deja ete determine avant, sous peine de refus.
** L'utilisateur disposera de 3 essais au maximum, pour determiner ce bareme
** composant sinon la creation en cours se verra automatiquement interrompue
** et le retour vers le menu se produira.

```
MOVE 00 TO NUM_PROC.MOVE 61 TO NUM_HESS.PERFORM CALL_ES1.  
MOVE 09 TO NUM_PROC PERFORM CALL_MBAR2.  
IF EXISTEC = 0 MOVE 47 TO NUM_HESS MOVE 00 TO NUM_PROC  
PERFORM CALL_ES1 MOVE 13 TO NUM_PROC  
PERFORM CALL_ES1 GO TO PROG_TEST.  
MOVE 16 TO NUM_HESS.  
MOVE 00 TO NUM_PROC.  
PERFORM CALL_ES1.  
MOVE 06 TO NUM_PROC.PERFORM CALL_ES1.  
MOVE 05 TO NUM_PROC.  
PERFORM CALL_MBAR2.MOVE EXISTEC TO EXCPT1.  
IF EXCPT1 = 0 SUBTRACT 1 FROM I  
ADD 1 TO I  
IF I>2 MOVE 80 TO NUM_HESS  
MOVE 00 TO NUM_PROC  
PERFORM CALL_ES1  
MOVE 13 TO NUM_PROC  
PERFORM CALL_ES1  
GO TO PROG_TEST  
END-IF  
MOVE 17 TO NUM_HESS  
MOVE 00 TO NUM_PROC  
PERFORM CALL_ES1  
MOVE 13 TO NUM_PROC PERFORM CALL_ES1  
MOVE 00 TO NUM_PROC MOVE 61 TO NUM_HESS  
PERFORM CALL_ES1  
END-IF.
```

*

CONTROL_EXIST_BAR.

** Cette primitive s'occupe de lire un libelle de bareme a l'ecran, d'en
** controler l'existence (sous peine de refus) et en cas favorable de
** controler si l'utilisateur qui travaille a acces a ce bareme bareme (sous
** peine de refus egalement).

MOVE 06 TO NUM_PROC.PERFORM CALL_ES1.

MOVE 05 TO NUM_PROC.

PERFORM CALL_IBAR2.

MOVE EXISTEC TO EXCPT1.

IF EXCPT1 = 0 MOVE 17 TO NUM_MESS

MOVE 00 TO NUM_PROC

PERFORM CALL_ES1

MOVE 13 TO NUM_PROC

PERFORM CALL_ES1 GO TO PROG_TEST

ELSE MOVE 1 TO FINI

MOVE 02 TO NUM_PROC

MOVE NUM_BAR TO ID_BAR

PERFORM CALL_SEC

IF EXISTE=0

MOVE 74 TO NUM_MESS MOVE 00 TO NUM_PROC PERFORM CALL_ES1

MOVE 13 TO NUM_PROC PERFORM CALL_ES1 GO TO PROG_TEST

END-IF.

NIVEAU4 SECTION.

VALID_VAL.

** Dans la determination d'une valeur de code composante, cette primitive s'oc
** cupe de lire et de valider cette valeur. Cette validation controle en fait
** l'existence, et l'actualite de la valeur (refus si pas valable)
** L'utilisateur dispose de 3 essais pour localiser cette valeur sinon la
** creation sera interrompue et l'on retournera au menu principal.

IF N = 1 MOVE 62 TO NUM_MESS

MOVE IDENT_BAR(J) TO NUM_BAR.

IF N = 2 MOVE 59 TO NUM_MESS

MOVE IDENT_BAR(J) TO NUM_BAR.

MOVE 00 TO NUM_PROC.PERFORM CALL_ES1.

MOVE 02 TO NUM_PROC.

PERFORM CALL_IBAR2.

MOVE 11 TO NUM_PROC.

PERFORM CALL_ES2 VARYING I FROM 1 BY 1 UNTIL I>NLC.

DISPLAY "".

MOVE 00 TO NUM_PROC.MOVE 44 TO NUM_MESS.PERFORM CALL_ES1.

MOVE 13 TO NUM_PROC.

PERFORM CALL_ES1.

*


```

MOVE NUM_BAR TO IDVC_PART1.
MOVE VAL_CODE TO IDVC_PART2.
MOVE 03 TO NUM_PROC.
PERFORM CALL_IBAR2.
IF EXISTEC = 1
    CALL "TIMER" USING BY REFERENCE TIME_FIELD, HOUR_FIELD
    CALL "COMPARE_DATE" USING BY REFERENCE
                                DTC_DB, TIME_FIELD, ETAT1
    CALL "COMPARE_DATE" USING BY REFERENCE
                                TIME_FIELD, DTC_FN, ETAT2

    IF ETAT1 = "SUP"
        MOVE 66 TO NUM_MESS
        MOVE 00 TO NUM_PROC
        PERFORM CALL_ES2
    END-IF
    IF ETAT2 = "SUP"
        MOVE 67 TO NUM_MESS
        MOVE 00 TO NUM_PROC
        PERFORM CALL_ES2
    END-IF
    MOVE "" TO ETAT3
    IF ETAT1 = "EGA"
        CALL "COMPARE_HEURE" USING BY REFERENCE HRC_DB, HOUR_FIELD, ETAT3
        IF ETAT3="SUP" MOVE 66 TO NUM_MESS
                        MOVE 00 TO NUM_PROC
                        PERFORM CALL_ES2
        END-IF
    END-IF
    IF ETAT1 NOT EQUAL "SUP" AND ETAT2 NOT EQUAL "SUP"
        AND ETAT3 NOT EQUAL "SUP"
        MOVE 1 TO FINI
    ELSE MOVE 13 TO NUM_PROC PERFORM CALL_ES2 MOVE 0 TO FI
ELSE MOVE 40 TO NUM_MESS MOVE 00 TO NUM_PROC
    PERFORM CALL_ES2
    ADD 1 TO Z
    IF Z > 02 MOVE 80 TO NUM_MESS
                MOVE 00 TO NUM_PROC
                PERFORM CALL_ES1
                MOVE 13 TO NUM_PROC
                PERFORM CALL_ES1
                GO TO PROG_TEST
    END-IF
    MOVE 13 TO NUM_PROC
    PERFORM CALL_ES2.

```

EFFACE_TAB_IDS.

** Cette primitive a pour but de remettre a zero le tableau TAB_IDS.
 ** On a ici le corps de la boucle qui le fait pour la composante I.

MOVE 0 TO IBAR(I).

 *

SORTIR_NON_BAR.

** Cette primitive doit sortir a l'ecran les libelles du bareme dont l'identificateur se trouve dans NBAR(Z).

```
ADD 1 TO Z.
IF J=4 MOVE 1 TO EC.
IF Z=NC MOVE 1 TO FINI MOVE 1 TO EC.
MOVE 02 TO NUL_PROC.
MOVE NBAR(Z) TO ID_BAR.
PERFORM CALL_NBAR1.
MOVE 11 TO NUL_PROC.
PERFORM CALL_ES1 VARYING I FROM 1 BY 1 UNTIL I>NL.
DISPLAY "".
```

DOUBLE.

** Cette primitive doit controler si dans le tableau IDENT_BAR, les elements de position I et J sont identiques. Ceci dans le but de controler si l'identificateur de bareme que l'on vient de determiner n'avait pas deja ete determine et entre precedemment dans le tableau.

```
IF IDENT_BAR(I)=IDENT_BAR(J) MOVE 1 TO EXCPT
                                MOVE 1 TO FINI.
IF Z=J MOVE 1 TO FINI.
ADD 1 TO J.
```

SORTIR_BAR_COMP.

** Cette primitive a pour but de sortir a l'ecran l'en-tete du bareme composant dont l'identificateur se trouve dans IDENT_BAR(Z).

```
MOVE 01 TO NUL_PROC.PERFORM CALL_ES2.
MOVE 02 TO NUL_HESS.
MOVE 00 TO NUL_PROC.
PERFORM CALL_ES2.
MOVE IDENT_BAR(Z) TO NUL_BAR.
MOVE 02 TO NUL_PROC.
PERFORM CALL_NBAR2.
MOVE 14 TO NUL_PROC.PERFORM CALL_ES2.
MOVE 05 TO NUL_HESS.
MOVE 00 TO NUL_PROC.
PERFORM CALL_ES2.
MOVE 13 TO NUL_PROC.PERFORM CALL_ES2.
```

*

SORTIR_REL_COMP.

** Cette primitive a pour but de sortir a l'ecran la valeur composante et ses
** renseignements (dont son identificateur se trouve dans NREL(Z)). Cette
** valeur faisant partie du bareme dont l'identifiant se trouve dans
** IDENT_PAF(Z).

MOVE 01 TO NUM_PROC.PERFORM CALL_ES1.
MOVE 46 TO NUM_MESS.
MOVE 00 TO NUM_PROC.
PERFORM CALL_ES1.
MOVE IDENT_PAF(Z) TO NUM_BAR.MOVE 02 TO NUM_PROC.
PERFORM CALL_NBAR2.MOVE 11 TO NUM_PROC.
PERFORM CALL_ES2 VARYING I FROM 1 BY 1 UNTIL I>NLC.
MOVE NREL(Z) TO ID_REL.
MOVE 08 TO NUM_PROC.
PERFORM CALL_NBAR1.
MOVE IDV_PART2 TO VAL_CODE.
MOVE 20 TO NUM_PROC.PERFORM CALL_ES1.
MOVE 05 TO NUM_MESS.
MOVE 00 TO NUM_PROC.
PERFORM CALL_ES1.
MOVE 13 TO NUM_PROC.PERFORM CALL_ES1.

COPIE_LIB_TABLEAU.

** Cette primitive doit, dans le but de la sortie d'un tableau de valeurs a
** l'ecran, aller mettre le iier libelle du bareme, dont l'identifiant est
** dans IDENT_BAR(I), dans LIBS(Z).

MOVE IDENT_BAR(Z) TO NUM_BAR.MOVE 02 TO NUM_PROC.
PERFORM CALL_NBAR2.
MOVE LIBELLE(I) TO LIBS(Z).

COPIE_REL_RELC.

** Cette primitive recopie la composante J du tableau TAB_RELS dans la compo-
** sante J du tableau TAB_RELC. Ceci afin de recopier l'entierete de TAB_RELS
** dans TAB_RELC.

MOVE NREL(J) TO NREL(J).

COPIE_LIBC_LIBS.

** Cette primitive recopie la composante Z du tableau TAB_LIBC dans la compo-
** sante Z du tableau TAB_LIBS. Ceci afin de recopier l'entierete de TAB_LIBC
** dans TAB_LIBS.

MOVE LIBELLE(Z) TO LIBELLE(Z).
MOVE REG_LINGC(Z) TO REG_LING(Z).

*

CALL_IBAR1.

** Cette primitive procede a l'appel du module BAREME sur le premier jeux
** de variables.

CALL "MODULE_BAREME" USING BY REFERENCE
NUM_PROC, ID_BAR, EXISTE, NL, LIBS_BAR, DT_CR, TP_VAL,
LG_VAL, NB_VERS, DR_RET, NB, TAB_BARS, NC,
TAB_IDS, NV, TAB_RELS, LIBLU, ID_REL, DT_DB, HR_DB, DT_FN,
TABLEAU_VALS, NLIG, TERMI, W, CLE.

CALL_MBAR2.

** Cette primitive procede a l'appel du module BAREME sur le deuxieme jeux
** de variables.

CALL "MODULE_BAREME" USING BY REFERENCE
NUM_PROC, NUM_BAR, EXISTEC, NLC, LIBC_BAR, DTC_CR, TPC_VAL,
LGC_VAL, NVC_VERS, DRC_RET, NRC, TAB_BARC, NCC,
TAB_IDC, NVC, TAB_RELC, LIBLU, NUM_REL, DTC_DB, HRC_DB, DTC_FN,
TABLEAU_VALS, NLIG, TERMI, W, CLE.

CALL_ES1.

** Cette primitive procede a l'appel du module ENTREES/SORTIES sur le
** premier jeux de variables.

CALL "MODULE_ENTREE_SORTIE" USING BY REFERENCE
NUM_PROC, NUM_MESS, ID_BAR, LIBS_BAR, NL,
TAB_BARS, NB, NBP, DT_CR, TP_VAL, LG_VAL,
DR_VERS, DR_RET, LIBLU, EXISTE, FINI, I,
DT_MESS, DT_DB, HR_DB, DT_FN, VAL_CODE,
ID_REL, NV, TAB_RELS, REP, TABLEAU_VALS, NLIG,
TABLEAU_LIBS, LIBBAR, ID_UTIL.

CALL_ES2.

** Cette primitive procede a l'appel du module ENTREES/SORTIES sur le
** deuxieme jeux de variables.

CALL "MODULE_ENTREE_SORTIE" USING BY REFERENCE
NUM_PROC, NUM_MESS, NUM_BAR, LIBC_BAR, NLC,
TAB_BARC, NRC, NBP, DTC_CR, TPC_VAL, LGC_VAL,
NRC_VERS, DRC_RET, LIBLU, EXISTEC, FINI, I,
DTC_MESS, DTC_DB, HRC_DB, DTC_FN, VAL_CODE,
NUM_REL, NVC, TAB_RELC, REP, TABLEAU_VALS, NLIG,
TABLEAU_LIBS, LIBBAR, ID_UTIL.

*

CALL-SEC.

** Cette primitive procede a l'appel du module SECURITE sur le
** premier jeux de variables.

CALL "MODULE_SECURITE" USING BY REFERENCE
NUM_PROC, ID_UTIL, ID_BAR, TAB_IDS, NC, EXISTE.

CALL-SEC2.

** Cette primitive procede a l'appel du module SECURITE sur le
** deuxieme jeux de variables.

CALL "MODULE_SECURITE" USING BY REFERENCE
NUM_PROC, ID_UTIL, NUM_BAR, TAB_IDC, NCC, EXISTEC.

CALL-CONTROL_VAL.

** Cette primitive procede a l'appel du module "CONTROL_CODE".

CALL "CONTROL_CODE" USING BY REFERENCE
VAL_CODE, LG_VAL, TE_VAL, ETAT1, ETAT2.

Programme BARPROG.COB

IDENTIFICATION DIVISION.

PROGRAM-ID. GESTION-BAREME-PAR-PROGRAMME.

AUTHOR. E EVRARD.

DATE-WRITTEN. 01/07/1986.

**
** GESTION DE BAREMES PAR PROGRAMMES **
**

** Ce module a pour but de repondre aux 2 questions que l'on peut
** poser par Programmes:
** 1 La recherche selon critere.
** 2 Obtenir les libelles d'une valeur de code.
**
** Si on ne peut repondre, le module renvoie un numero d'erreur dont la
** signification peut etre lue dans le fichier ESSAI.COB (primitive LEGENDE).
**

** Question 1 Recherche selon critere.

** -----

** Arguments d'entree:

**
** NOM_UTIL : nom de l'utilisateur
** PWD_UTIL : mot de passe de l'utilisateur
** OPT_MENU : numero de la question posee (ici 1)
** LIBLU : libelle du bareme sur lequel on desire effectuer la recherche
** TAB_CRIT : tableau contenant le critere de recherche
** LCRIT : longueur de ce tableau
**

** Arguments de sortie:

**
** TABLEAU_VALS : tableau contenant les valeurs qui resultent de la recherche
** NLIG : longueur de ce tableau
** N : numero du tableau resultant
** CLE : identificateur dont la gestion de baremes a besoin pour con-
** tiner sa recherche
** TERM : indicateur qui nous dit oui(=1) ou non(=0) la recherche est
** terminee
** LIBBAR : libelle du bareme objet de la recherche
** TABLEAU_LIPS : tableau contenant les libelles des colonnes (c-a-d des
** baremes composants) du tableau des valeurs a sortir a l'ecra
** N : longueur de ce tableau
** NUM_ERP : numero de l'erreur si l'on ne peut satisfaire la demande
** (il vaut 0 s'il peut le faire)
**

** Question 2 Obtenir les libelles d'une valeur de code.

** -----

** Arguments d'entree:

**

** NOM_UTIL : nom de l'utilisateur

** PSWD_UTIL : mot de passe de l'utilisateur

** OPT_MENU : numero de la question posee (ici 2)

** LIBLU : libelle du bareme auquel on veut acceder

** VAL_CODE : valeur de code dont on desire les renseignements

**

** Arguments de sortie:

**

** DT_CR : date de creation de la valeur de code

** DT_DB : date de debut d'utilisation de la valeur de code

** HR_DB : heure " " " "

** DT_FN : date de fin " " " "

** LIBS_BAR : tableau des libelles associes a cette valeur de code

** NL : longueur de ce tableau

** TAB_RELS : tableau des valeurs de code composantes

** NV : longueur de ce tableau

** NUM_ERP : numero de l'erreur si on ne peut satisfaire la demande.

** (il vaut 0 si on peut le faire)

**

ENVIRONMENT DIVISION.

CONFIGURATION SECTION.

SOURCE-COMPUTER. vax_11.
OBJECT-COMPUTER. vax_11.

DATA DIVISION.

WORKING-STORAGE SECTION.

01 I PIC 9(1).
01 ETAT1 PIC X(3).
01 ETAT2 PIC X(3).
01 ETAT3 PIC X(3).
01 TIME_FIELD PIC X(10).
01 HOUR_FIELD PIC X(5).
01 STOPPE PIC 9(1).
01 EXISTE PIC 9(1).
01 EXISTEC PIC 9(1).
01 EXCEP1 PIC 9(1).
01 ID_BAR PIC X(6).
01 NUM_BAR PIC X(6).
01 NLC PIC 9(2).
01 LIBC_BAR.
02 ELEMENT OCCURS 1 TO 10 TIMES DEPENDING ON NLC.
03 LIBELLEC PIC X(50).
03 REC_LINGC PIC X(2).
01 PTC_CR PIC X(10).
01 TP_VAL PIC X(2).
01 PPC_VAL PIC X(2).
01 LG_VAL PIC X(2).
01 LGC_VAL PIC X(2).
01 LG_VAL_CAL PIC 9(2).
01 NB_VERS PIC X(2).
01 NBC_VERS PIC X(2).
01 DR_RET PIC X(4).
01 DRC_RET PIC X(4).
01 NBC PIC 9(2).
01 NBP PIC 9(2).
01 TAB_IDS.
02 NBAR PIC X(6) OCCURS 99 TIMES.
01 TAB_IDC.
02 NBART PIC X(6) OCCURS 99 TIMES.
01 TAB_BARS.
02 ELEMENT OCCURS 1 TO 10 TIMES DEPENDING ON NB.
03 IDENT_BAR PIC X(6).
03 ROLE_BAR PIC X(50).
03 DUR PIC 9(4).

*

```

01 TAB_BARC.
  02 ELEMENT OCCURS 1 TO 10 TIMES DEPENDING ON NB.
    03 IDENT_BARC PIC X(6).
    03 POLE_BARC PIC X(50).
    03 DUPE PIC 9(4).
01 TAB_BAPP.
  02 IDENT_BAPP;PIC X(6);OCCURS 10 TIMES.
01 Z PIC 9(2).
01 I PIC 9(2).
01 J PIC 9(2).
01 HC PIC 9(2).
01 HCC PIC 9(2).
01 HVC PIC 9(2).
01 TAB_RELC.
  02 ELEMENT OCCURS 10 TIMES.
    03 PRELC PIC X(12).
    03 PRELC PIC X(50).
01 ID_VALEUR.
  02 IDV_PART1 PIC X(6).
  02 IDV_PART2 PIC X(6).
01 ID_REL ; REDEFINES ID_VALEUR ; PIC X(12).
01 IDC_VALEUR.
  02 IDVC_PART1 PIC X(6).
  02 IDVC_PART2 PIC X(6).
01 NUM_REL ; REDEFINES IDC_VALEUR ; PIC X(12).
01 DTC_DB PIC X(10).
01 NUM_PROC PIC 9(2).
01 HRC_DB PIC X(5).
01 DTC_FH PIC X(10).
01 FINI PIC 9(1).
01 ID_UTIL.
  02 NAME PIC X(10).
  02 DSMD PIC X(10).

```

*

LINKAGE SECTION.

*-----

```

01 NOM_UTIL PIC X(10).
01 PSUD_UTIL PIC X(10).
01 NUM_ERR PIC 9(2).
01 VAL_CODE PIC X(6).
01 NL PIC 9(2).
01 LIBS_BAR.
    02 ELEMENT OCCURS 1 TO 10 TIMES DEPENDING ON NL.
        03 LIBELLE PIC X(50).
        03 REC_LING PIC X(2).
01 DT_CR PIC X(10).
01 NB PIC 9(2).
01 LIBLU PIC X(50).
01 NV PIC 9(2).
01 TAB_RELS.
    02 ELEMENT OCCURS 10 TIMES.
        03 NREL PIC X(12).
        03 RREL PIC X(50).
01 TAB_CRIT.
    02 ELEMENT OCCURS 10 TIMES.
        03 TCR_BAR PIC X(6).
        03 TCR_VAL PIC X(6).
01 LCRIT PIC 9(2).
01 DT_DB PIC X(10).
01 NR_DB PIC X(5).
01 DT_FW PIC X(10).
01 TABLEAU_VALS.
    02 ELT1; OCCURS 08 TIMES.
        03 ELT2 PIC X(6); OCCURS 11 TIMES.
01 NLIG PIC 9(2).
01 TABLEAU_LIPS.
    02 LIPS; PIC X(12); OCCURS 10 TIMES.
01 LIBBAR PIC X(50).
01 TERMI PIC 9(1).
01 W PIC 9(2).
01 CLE PIC X(12).
01 OPT_MENU PIC 9(1).

```

*

PROCEDURE DIVISION USING OPT_MENU, NOM_UTIL, PSWD_UTIL, LIBLU, TAB_CRIT, LCRIT,
TABLEAU_VALS, NLIG, TERMI, CLE, N, LIBBAR, TABLEAU_LIBS, N
VAL_CODE, DT_CR, DT_DB, HR_DB, DT_FN, LIBS_BAR, NL, TAB_REI
HV, NUM_ERR.

NIVEAU1 SECTION.

PROG_IDENT.

** Cette primitive lance l'identification de l'utilisateur.
** Remarque: on en ressort pour se brancher vers le programme
** general uniquement si l'identification a reussi.

MOVE 0 TO STOPPE. MOVE NOM_UTIL TO NAME. MOVE PSWD_UTIL TO PSWD.
PERFORM IDENTIFIER.
GO TO PROG_TEST.

PROG_TEST.

** Programme general qui controle si l'argument OPT_MENU est bien compris ent
** 1 et 2, et s'il existe des baremes dans le systeme. En cas contraire, un
** numero d'erreur est renvoye sinon on se branche vers la primitive appropri
** selon la valeur de OPT_MENU.

IF OPT_MENU > 0 AND OPT_MENU < 3
MOVE 02 TO NUM_PROC PERFORM CALL_LIBAR1
IF EXISTE = 0 MOVE 12 TO NUM_ERR
EXIT PROGRAM
END-IF
ELSE MOVE 13 TO NUM_ERR
EXIT PROGRAM.
IF OPT_MENU = 2 PERFORM OBTENIR_LIBELLES_VALEUR.
IF OPT_MENU = 1 PERFORM RECHERCHE_CRITERE.

*

NIVEAU2 SECTION.

IDENTIFIEP.

** Ici on fait l'appel au module securite afin de voir si l'utilisateur
** existe. En cas echec, un numero d'erreur est renvoye.

MOVE 1 TO NUM_PROC. PERFORM CALL_SEC.
IF EXISTE = 0 MOVE 01 TO NUM_ERR
EXIT PROGRAM.

RECHERCHE_CRITERE.

** Cette primitive s'occupe de la recherche selon critere. Si le bareme
** demande est existant, s'il possede des valeurs, s'il est compose, et
** si le critere le recherche est correct, la recherche aura effectivement
** lieu. Selon les valeurs de W et CLE, le module BAREME sait quel tableau
** de valeurs il doit renvoyer. (C'est le premier ou alors les suivants
** a partir de la valeur de CLE). Le module BAREME nous renverra TERMI
** egale a 01 seulement si la recherche est terminee.

PERFORM INIT_TABL_LIBS VARYING I FROM 1 BY 1 UNTIL I>10.
PERFORM INIT_TABL_VALS VARYING I FROM 1 BY 1 UNTIL I>8.
PERFORM COPIE_CRIT VARYING I FROM 1 BY 1 UNTIL I>LCRIT.
PERFORM CONTROL_EXIST_BAR.
MOVE 10 TO NUM_PROC. MOVE NUM_BAR TO ID_BAR.
PERFORM CALL_MBAR1.
IF EXISTE = 0 MOVE 04 TO NUM_ERR
EXIT PROGRAM.
MOVE 02 TO NUM_PROC. PERFORM CALL_MBAR1.
MOVE LIB00 TO LIBBAR.
IF NB>0
PERFORM CONTROL_BAR_CRIT VARYING J FROM 1 BY 1 UNTIL J>LCRIT
PERFORM AVDIR_LIB VARYING I FROM 1 BY 1 UNTIL I>NB
MOVE LCRIT TO W
PERFORM CONTROL_REL_CRIT VARYING J FROM 1 BY 1 UNTIL J>W
IF W=00 MOVE 01 TO J ELSE ADD 1 TO W END-IF
MOVE 0 TO TERMI
MOVE 12 TO NUM_PROC
PERFORM CALL_MBAR1
IF TERMI=01 MOVE SPACES TO CLE MOVE 00 TO W END-IF
EXIT PROGRAM
ELSE MOVE 05 TO NUM_ERR
EXIT PROGRAM.

*

OBTENIR_LIBELLES_VALEUR.

** Cette primitive s'occupe de la question 2. Si le bareme demande existe,
** s'il y a des valeurs et si la valeur de code donnee existe, on obtiendra
** effectivement la reponse sinon un numero d'erreur sera renvoye.

```
PERFORM CONTROL_EXIST_BAR.  
MOVE 10 TO NUM_PROC.MOVE NUM_BAR TO ID_BAR.  
PERFORM CALL_HBAR1.  
IF EXISTE = 0 MOVE 04 TO NUM_ERR  
EXIT PROGRAM.  
MOVE VAL_CODE TO IDV_PART2.  
MOVE NUM_BAR TO IDV_PART1.  
MOVE 08 TO NUM_PROC.PERFORM CALL_HBAR1.  
IF EXISTE=0  
MOVE 11 TO NUM_ERR  
EXIT PROGRAM  
END-IF.  
EXIT PROGRAM.
```

NIVEAUX SECTION.

COPIE_CRIT.

** Cette primitive recopie la partie I du critere de recherche dans la
** position I de TAB_RELS. Le module BAREME doit en effet recevoir le
** critere de recherche dans TAB_RELS

```
MOVE TCP_BAR(I) TO IDV_PART1.  
MOVE TCP_VAL(I) TO IDV_PART2.  
MOVE ID_VALEUR TO NREL(I).
```

AVOIR_LIB.

** Cette primitive accede au premier libelle du bareme (dont l'identifiant
** se trouve dans IDENT_BAR(I)) et le met dans LIBS(I).

```
MOVE IDENT_BAR(I) TO NUM_BAR.  
MOVE 02 TO NUM_PROC.  
PERFORM CALL_HBAR2.  
MOVE LIBELLEC(1) TO LIBS(I).
```

INIT-TABL-LIBS.

** Initialisation de TAB-LIBS.

MOVE SPACES TO LIBS(I).

INIT-TABL-VALS.

** Initialisation de TABLEAU-VALS.

PERFORM INIT-LIGNE VARYING J FROM 1 BY 1 UNTIL J>11.

INIT-LIGNE.

** Initialisation de TABLEAU-VALS.

MOVE SPACES TO ELT2(I,J).

CONTROL-REFL-CRIT.

** Cette primitive doit valider la valeur de code se trouvant dans TCR_VAL(J).
** le numero de bareme associe est dans TCR_BAR(J). Cette validation comprend
** le test d'existence et d'actualite de la valeur.

MOVE 0 TO FINI.
MOVE TCR_BAR(J) TO NUM_BAR.
MOVE TCR_VAL(J) TO VAL_CODE.
PERFORM VALID_VAL UNTIL FINI = 1.
MOVE NUM_REL TO NREL(J).

CONTROL-BAR-CRIT.

** Cette primitive doit controler si le bareme dont l'identificateur est dans
** TCR_BAR(J) fait partie du tableau des baremes composants TAB_BARS. Sinon
** le critere de recherche est incorrect et un numero d'erreur est renvoye.

MOVE 0 TO EXISTE.
PERFORM CONTROL_VALID_CRIT_BAR VARYING I FROM 1 BY 1 UNTIL (I>UB)
OR (EXISTE=1).
IF EXISTE=0
MOVE 06 TO NUM_ERR
EXIT PROGRAM
END-IF.

*

CONTROL_EXIST_BAR.

** Cette primitive doit verifie si un bareme existe et dans l'affirmative
** doit nous dire si l'utilisateur y a acces ou pas

```
MOVE 05 TO NUM_PROC.  
PERFORM CALL_IBAR2.  
MOVE EXISTEC TO EXCPT1.  
IF EXCPT1=0 MOVE 02 TO NUM_ERR  
EXIT PROGRAM  
ELSE  
MOVE 02 TO NUM_PROC  
MOVE NUM_BAR TO ID_BAR  
PERFORM CALL_SEC  
IF EXISTE=0  
MOVE 03 TO NUM_ERR  
EXIT PROGRAM  
END-IF.
```


NIVEAU4 SECTION.

CONTROL_VALID_CRIT_BAR.

** Corp de boucle de la primitive CONTROL_BAR_CRIT

```
IF TCF_BAR(J)=IDENT_BAR(I) MOVE 1 TO EXISTE.
```

*

VALID_VAL.

** Cette primitive effectue le controle d'existence et d'actualite de VAL_CODE
** dans le bareme d'identifiant NUM_BAR.

MOVE NUM_BAR TO IDVC_PART1.

MOVE VAL_CODE TO IDVC_PART2.

MOVE 08 TO NUM_PROC.

PERFORM CALL_HBAR2.

IF EXISTEC = 1

CALL "TIMER" USING BY REFERENCE TIME_FIELD, HOUR_FIELD

CALL "COMPARE-DATE" USING BY REFERENCE

DTC_DB, TIME_FIELD, ETAT1

CALL "COMPARE-DATE" USING BY REFERENCE

TIME_FIELD, DTC_FN, ETAT2

IF ETAT1 = "SUP"

MOVE 07 TO NUM_ERR

EXIT PROGRAM

END-IF

IF ETAT2 = "SUP"

MOVE 08 TO NUM_ERR

EXIT PROGRAM

END-IF

MOVE "" TO ETAT3

IF ETAT1 = "EGA"

CALL "COMPARE-HEURE" USING BY REFERENCE HRC_DB, HOUR_FIELD, ETAT3

IF ETAT3="SUP"

MOVE 07 TO NUM_ERR

EXIT PROGRAM

END-IF

END-IF

IF ETAT1 NOT EQUAL "SUP" AND ETAT2 NOT EQUAL "SUP"

AND ETAT3 NOT EQUAL "SUP"

MOVE 1 TO FINI

ELSE MOVE 0 TO FINI

ELSE MOVE 02 TO NUM_ERR

EXIT PROGRAM.

CALL_HBAR1.

** Appel au module BAREME sur le premier jeux de variables.

CALL "MODULE_BAREME" USING BY REFERENCE

NUM_PROC, ID_BAR, EXISTEC, NL, LIBS_BAR, DT_CR, TP_VAL,

LC_VAL, NB_VERS, DR_RET, NB, TAB_BARS, NC,

TAB_IDS, NV, TAB_RELS, LIBLU, ID_REL, DT_DB, HR_DB, DT_FN,

TABEAN_VALS, NLIG, TERMI, W, CLE.

*

CALL_NBAR?

** Appel au module BARENE sur le deuxieme jeux de variables.

CALL "MODULE_BARENE" USING BY REFERENCE
HUN_PROC, HUN_BAR, EXISTEC, HLC, LIBC_BAR, DTC_CR, TPC_VAL,
LCC_VAL, HBC_VERS, DRC_RET, HBC, TAB_BARC, NCC,
TAB_IDC, HVC, TAB_RELC, LIBDU, HUN_REL, DTC_DB, HRC_DB, DTC_FN,
TABLEAU_VALS, NLIG, TERMI, W, CLE.

CALL_SEC.

** Appel au module SECURITE.

CALL "MODULE_SECURITE" USING BY REFERENCE
HUN_PROC, ID_UTIL, ID_BAR, TAB_IDS, NC, EXISTE.

3/ Niveau du bas.

Programme BAREME.COB

IDENTIFICATION DIVISION.

PROGRAM-ID. MODULE_BAREME.

AUTHOR. E EMPARD.

DATE-WRITTEN. 14/04/1986.

**
**
**
**
**
**

** Dans ce module, se trouve regroupe l'ensemble des primitives qui travail-
** lent sur la structure de donnees qu'on a developpee. Seul ce programme
** connaît la structure du schéma des baremes de notre BD. Ce programme doit
** répondre aux demandes des programmes appelant. En voici la liste:
** 1) Créer un en-tête de bareme;
** 2) Obtenir le bareme dont l'identifiant est ID_BAR;
** 3) Existe-t'il un bareme dont l'identificateur est ID_BAR?
** 4) Renvoyer tous les identificateurs de baremes;
** 5) Voir si un libelle de bareme existe;
** 6) Renvoyer les identificateurs des baremes qui admettent le bareme d'iden-
** tifiant ID_BAR comme composant;
** 7) Créer une valeur le baremes;
** 8) Obtenir une valeur le baremes dont l'identificateur est ID_REL;
** 9) Voir si un bareme contient des valeurs;
** 10) Voir si le systeme comprend déjà un bareme;
** 11) Donner un tableau le valeurs du bareme dont le numero est ID_BAR a
** partir d'une certaine valeur memorisee;
** 12) Donner un tableau le valeurs d'actualite du bareme compose dont le
** numero est ID_BAR. Les lignes le ce tableau doivent contenir les valeurs
** de code memorisees dans TAB_RELS (contenant le critere de recherche);
** 13) Supprimer les valeurs perimees du bareme de numero ID_BAR;
** 14) Supprimer les baremes perimes du systeme.

ENVIRONMENT DIVISION.

CONFIGURATION SECTION.

SOURCE-COMPUTER. vax_11.
OBJECT-COMPUTER. vax_11.

*

DATA DIVISION.

SUB-SCHEMA SECTION.

DB DEFAULT SUBSCHEMA WITHIN GESTBAR FOR "GESTBAR.ROO".
LD KEEP-LIST.

WORKING-STORAGE SECTION.

01 Q PIC X(1).
01 DT_SYS PIC X(10).
01 DT_PE PIC X(10).
01 HEURE PIC X(5).
01 ETAT1 PIC X(3).
01 ETAT2 PIC X(3).
01 ETAT3 PIC X(3).
01 STOPPE PIC 9(1).
01 I ; PIC 9(2).
01 J ; PIC 9(2).
01 Z ; PIC 9(2).
01 FINI ; PIC 9(1).
01 COM ; PIC X(12).
01 ELT.
 02 IDB PIC X(6).
 02 IDV PIC X(6).
01 IDBV ; REDEFINES ELT ; PIC X(12).
01 FRUC.
 02 IDBT PIC X(6).
 02 IDVT PIC X(6).

LINKAGE SECTION.

01 ID_BAR PIC X(6).
01 HB PIC 9(2).
01 HL PIC 9(2).
01 LIBS_BAR.
 02 ELEMENT OCCURS 1 TO 10 TIMES DEPENDING ON HL.
 03 LIPELLE PIC X(50).
 03 REC_LING PIC X(2).
01 DT_CR PIC X(10).
01 TP_VAL PIC X(2).
01 LG_VAL PIC X(2).
01 NB_VERS PIC X(2).
01 DR_RET PIC X(1).
01 TAB_BAPS.
 02 ELEMENT OCCURS 1 TO 10 TIMES DEPENDING ON HB.
 03 IDENT_PAR PIC X(6).
 03 ROLE_BAR PIC X(50).
 03 DUR PIC 9(1).
01 EXISTE PIC 9(1).
01 NUM_PROC PIC 9(2).


```

01 LIBLU PIC X(50).
01 NC PIC 9(2).
01 TAB_IDS.
    02 NBAR PIC X(6) OCCURS 22 TIMES.
01 NV PIC 9(2).
01 TAB_RELS.
    02 ELEMENT OCCURS 20 TIMES.
        03 RPFL PIC X(12).
        03 RPFL PIC X(50).
01 ID_REL PIC X(12).
01 DT_DB PIC X(10).
01 HR_DB PIC X(5).
01 DT_FN PIC X(10).
01 TABLEAU_VALS.
    02 ELT1 OCCURS 8 TIMES.
        03 ELT2 PIC X(6); OCCURS 11 TIMES.
01 NLIG PIC 9(2).
01 TERMI PIC 9(1).
01 W PIC 9(2).
01 CLE PIC X(12).

```

```

*****
PROCEDURE DIVISION USING NUM_PROC, ID_BAR, EXISTE,
    NL, LIBS_BAR, DT_CR,
    TP_VAL, LG_VAL, NR_VERS,
    DR_RET, NB, TAB_BARS,
    NC, TAB_IDS, NV, TAB_RELS, LIBLU,
    ID_REL, DT_DB, HR_DB, DT_FN,
    TABLEAU_VALS, NLIG, TERMI, W, CLE.
*****

```

DECLARATIVES.

```

** Ce primitive est appelee automatiquement par le systeme si une erreur
** d'execution concernant la base de donnees arrive. C'est ce qu'on appelle
** une procedure d'exception.

```

```

100-DATABASE-EXCEPTION SECTION.
    USE FOR DB-EXCEPTION ON OTHER.
100-PROCEDURE.
    DISPLAY "DATABASE EXCEPTION CONDITION".
    PERFORM 150-DISPLAY-MESSAGE.

150-DISPLAY-MESSAGE.
    CALL "DB13SIGNAL".
    ROLLBACK.
    STOP RUN.

```

END DECLARATIVES.

```

*****
*****
*

```

NIVEAU1 SECTION.

APPEL_PROCEDEURE.

** Programme general qui identifie selon la valeur de NUM_PROC la demande
** auquel il faut repondre.

```
IF NUM_PROC = 01 PERFORM CR_TETE_BAR.  
IF NUM_PROC = 02 PERFORM OBTENIR_BAR.  
IF NUM_PROC = 03 PERFORM EXIST_BAR.  
IF NUM_PROC = 04 PERFORM OBTENIR_ALL_IDBAR.  
IF NUM_PROC = 05 PERFORM EXIST_LIB_BAR.  
IF NUM_PROC = 06 PERFORM OBTENIR_COMPOSE.  
IF NUM_PROC = 07 PERFORM CR_VAL_BAR.  
IF NUM_PROC = 08 PERFORM OBTENIR_VAL_BAR.  
IF NUM_PROC = 09 PERFORM PAS_DE_BAREMES.  
IF NUM_PROC = 10 PERFORM PAS_DE_VALEURS.  
IF NUM_PROC = 11 PERFORM GIVE_HT_VALS.  
IF NUM_PROC = 12 PERFORM RECH_HT_VALS.  
IF NUM_PROC = 13 PERFORM SUPPRESSION_VALEURS.  
IF NUM_PROC = 14 PERFORM SUPPRESSION_BAREMES.
```


NIVEAU2 SECTION.

SUPPRESSION_VALEURS.

** Cette primitive supprime les valeurs perimees dans le bareme de numero
** ID_BAR.

```
READY PROTECTED UPDATE.  
MOVE 0 TO FINI.  
PERFORM SUPP_VAL UNTIL FINI=1.  
COMMIT.  
EXIT PROGRAM.
```

*

SUPPRESSION_BAREMES.

** Cette primitive supprime le bareme dont le numero est ID_BAR.

```
READY PROTECTED UPDATE.  
MOVE ID_BAR TO NR_BAR.  
FETCH FIRST BAREME USING NR_BAR.  
ERASE BAREME.  
COMMIT.  
EXIT PROGRAM.
```

RECH-UT_VALS.

** Cette primitive nous donne le tableau suivant de valeurs d'actualite dans 1
** recherche selon critere. Si ce n'est pas le premier, on retablit d'abord
** l'enlroit ou l'on est arrive dans l'accès aux articles REL_BAR_CODE.

```
READY CONCURRENT RETRIEVAL.  
IF N>1 MOVE CLE TO NRBAR_VALCODE  
    FIND NEXT REL_BAR_CODE USING NRBAR_VALCODE  
    MOVE " " TO REL_COMP_LR  
    PERFORM RESTAURE UNTIL REL_COMP_LR=NREL(1).  
PERFORM OUT_VAL VARYING I FROM 1 BY 1 UNTIL I>8 OR TERM=1.  
IF NLIC<8 MOVE " " TO CLE.  
ROLLBACK.  
EXIT PROGRAM.
```

CR_TETE_BAR.

** Cette primitive cree un nouvel en-tete de bareme a partir des arguments
** fournis en entree.

```
READY PROTECTED UPDATE.  
MOVE ID_BAR TO NR_BAR.  
MOVE DT_CR TO DATE_CREATION OF BAREME.  
MOVE TP_VAL TO TYPE_VAL.  
MOVE LG_VAL TO LONG_VAL.  
MOVE NB_VERS TO NBR_VERS_GARD.  
MOVE DR_PRT TO DUR_RETENTION.  
STORE BAREME.  
PERFORM TRSF_LIG_CD VARYING I FROM 1 BY 1 UNTIL I>NL.  
PERFORM TRSF_LIGL_BAR VARYING I FROM 1 BY 1 UNTIL I>NB.  
COMMIT.  
EXIT PROGRAM.
```

*

CP_VAL_BAR.

** Cette primitive cree une valeur dans un bareme.

```
READY PROTECTED UPDATE.
MOVE ID_BAR TO NRBAR_REL.
MOVE ID_REL TO NRBAR_VALCODE.
MOVE DT_CR TO DATE_CREATION OF REL_BAR_CODE.
MOVE DT_DB TO DATE_DEB_UTIL.
MOVE HR_DB TO HEURE_DEB_UTIL.
MOVE DT_FH TO DATE_FIN_UTIL.
STORE REL_BAR_CODE.
PERFORM TRSF_LIEN_REL VARYING I FROM 1 BY 1 UNTIL I>NV.
PERFORM TRSF_LIB_VAL VARYING I FROM 1 BY 1 UNTIL I>NL.
COMMIT.
EXIT PROGRAM.
```

OBTENIR_BAR.

** Cette primitive fournit l'en-tete du bareme dont le numero est ID_BAR.

```
READY CONCURRENT RETRIEVAL.
PERFORM TETE_BAR.
ROLLBACK.
EXIT PROGRAM.
```

EXIST_LIB_BAR.

** Cette primitive controle si la valeur de l'argument LIBLI egale l'item
** VAL_LIB_BAR d'un des records LIB_BAR. Si c'est le cas, alors EXISTE est
** mis a 1 sinon 0.

```
READY CONCURRENT RETRIEVAL.
MOVE 1 TO EXISTE.
MOVE LIBLI TO VAL_LIB_BAR.
FIND NEXT LIB_BAR USING VAL_LIB_BAR
  AT END MOVE 0 TO EXISTE.
IF EXISTE = 1 FETCH OWNER WITHIN LIEN_BAR_LIB
  MOVE NR_BAR TO ID_BAR.
ROLLBACK.
EXIT PROGRAM.
```

*

OBTENIR_COMPOSE.

** Cette primitive trouve tous les records BARENE owner d'un chemin EST_COMP_DE
** contenant un record LIEN_BAR dont la valeur d'item NR_BAR_LB est ID_BAR, et
** memorise le NR_BAR de ce record BARENE.

```
READY CONCURRENT RETRIEVAL.  
MOVE 0 TO FINI.  
MOVE ID_BAR TO NR_BAR_LB.  
PERFORM TROUVE_LB_SEQ VARYING I FROM 1 BY 1 UNTIL FINI=1.  
COMPUTE NC = I - 2.  
ROLLBACK.  
EXIT PROGRAM.
```

OBTENIR_ALL_I0BAR.

** Cette primitive accede a tous les records BARENE du systeme et en memorise
** le NR_BAR.

```
READY CONCURRENT RETRIEVAL.  
MOVE 0 TO FINI.  
PERFORM TROUVE_I0BAR VARYING I FROM 1 BY 1 UNTIL FINI=1.  
COMPUTE NC = I - 2.  
ROLLBACK.  
EXIT PROGRAM.
```

EXIST_BAR.

** Cette primitive essaye d'accéder a un record BARENE dont le numero identi-
** fiant est ID_BAR. S'il en existe encore un, EXISTE est mis a 1 sinon a 0.

```
READY CONCURRENT RETRIEVAL.  
MOVE ID_BAR TO NR_BAR.  
MOVE 1 TO EXISTE.  
FIND NEXT BARENE USING NR_BAR  
AT END MOVE 0 TO EXISTE.  
ROLLBACK.  
EXIT PROGRAM.
```

*

OBTENIR_VAL_BAR.

** Cette primitive accede a l'article REL_BAR_CODE dont la cle est ID_REL.
** S'il existe, on remplit les arguments et on accede a ses libelles
** (articles LIB_VAL_CODE) et EXISTE est mis a un sinon a 0.

```
READY CONCURRENT RETRIEVAL.
MOVE ID_REL TO NRBAR_VALCODE.
MOVE 1 TO EXISTE.
FETCH NEXT REL_BAR_CODE USING NRBAR_VALCODE
  AT END MOVE 0 TO EXISTE.
IF EXISTE = 1 MOVE DATE_CREATION OF REL_BAR_CODE TO DT_CR
  MOVE DATE_DEB_UTIL TO DE_DB
  MOVE HEURE_DEB_UTIL TO HR_DB
  MOVE DATE_FIN_UTIL TO DT_FN
  MOVE 0 TO FINI
  PERFORM EXTR_LIB_VAL VARYING I FROM 1 BY 1
    UNTIL FINI=1
  COMPUTE UV = I - 2
  MOVE 0 TO FINI
  PERFORM EXTR_LIEN_REL VARYING I FROM 1 BY 1
    UNTIL FINI=1
  COMPUTE UV = I - 2.
ROLLBACK.
EXIT PROGRAM.
```

PAS_DE_BAREMES.

** Cette primitive tente d'accéder a un article BAREME quelconque pour voir
** s'il existe déjà un bareme dans le systeme.

```
READY CONCURRENT RETRIEVAL.
MOVE 1 TO EXISTE.
FIND FIRST BAREME
AT END MOVE 0 TO EXISTE.
ROLLBACK.
EXIT PROGRAM.
```

PAS_DE_VALEURS.

** Cette primitive essaye d'accéder a un article REL_BAR_CODE quelconque appa
** tenant au bareme de numero ID_BAR. S'il en existe un, EXISTE est mis a 1
** sinon 0.

```
READY CONCURRENT RETRIEVAL.
MOVE 1 TO EXISTE.
MOVE ID_BAR TO NRBAR_REL.
FIND FIRST REL_BAR_CODE USING NRBAR_REL
AT END MOVE 0 TO EXISTE.
ROLLBACK.
EXIT PROGRAM.
```

*

GIVE_HT_VALS.

** Cette primitive doit donner le tableau suivant de valeurs d'actualite du
** bareme de numero ID_BAR. Si ce n'est pas le premier, il faut recommencer
** l'accès ou on l'avait arrete c-a-d a l'article REL_BAR_CODE dont l'item
** NRBAR_VALCODE vaut CLE.

```
READY CONCURRENT RETRIEVAL.  
MOVE ID_BAR TO NRBAR_REL.  
IF W > 1 MOVE CLE TO NRBAR_VALCODE  
    FIND NEXT REL_BAR_CODE USING NRBAR_VALCODE.  
MOVE 0 TO STOPPE.  
PERFORM TROUVE_VAL_SEQ VARYING J FROM 1 BY 1 UNTIL STOPPE = 1.  
IF TERM = 0 MOVE NRBAR_VALCODE TO CLE  
ELSE MOVE " " TO CLE.  
ROLLBACK.  
EXIT PROGRAM.
```

NIVEAU3 SECTION.

SUPP_VAL.

** Sous primitive de SUPPRESSION VALEURS. Son but est de trouver, dans le bareme
** le numero ID_BAR, la valeur suivante qui n'est plus d'actualite et de la
** supprimer si elle est perimee.

```
MOVE 0 TO STOPPE.  
PERFORM BOUCLE_ACCES UNTIL STOPPE=1.  
IF FINI=0 PERFORM CONTROL_PERIME.
```

OBT_VAL.

** Cette primitive doit tenter de remplir le tableau TABLEAU_VALS (8 lignes max)
** avec des valeurs de baremes qui sont d'actualite et reliees au critere de
** recherche.

```
MOVE 0 TO FINI.  
PERFORM TROUVE_VAL_OK UNTIL FINI=1. (Vair 2 pages plus loin)  
IF I=0 MOVE 0 TO FINI.
```

*

RESTAURE.

** Sous primitive de RECH-HT-VALS. Acces a l'article LIEN_REL suivant dans
** le chemin LIEN_RBC.

FETCH NEXT LIEN_REL WITHIN LIEN_RBC.

*

TROUVE_VAL_OK.

** Cette primitive trouve un article REL_BAR_CODE owner d'un chemin LIEN_RBC
 ** dont un des membres LIEN_REL a une valeur d'item REL_COMP_LR valant HREL(1)
 ** et controle si cet article est d'actualite et relie aux autres HREL via les
 ** autres articles LIEN_REL dans le chemin LIEN_RBC. Si ces conditions sont
 ** satisfaites, on insere les valeurs d'item adequates dans TAB_VALS sinon
 ** l'article est rejete.

MOVE HREL(1) TO REL_COMP_LR.
 FETCH NEXT LIEN_REL USING REL_COMP_LR

AT END MOVE 1 TO FINI

MOVE 1 TO TERMI

COMPUTE NLIQ = I - 1.

IF TERMI = 0 (*Il y a-t-il encore une telle valeur?*)

KEEP CURRENT USING KEEP_LIST

FETCH OWNER WITHIN LIEN_RBC

MOVE HBAR_VALCODE TO IDBV

MOVE IDV TO ELG(1,11)

IF HBAR_REL = ID_BAR (*La valeur REL_BAR_CODE est-elle liée au barème que l'on désire?*)

MOVE HBAR_VALCODE TO CLE

CALL "TIMER" USING

BY REFERENCE DT_SYS, HEURE

CALL "COMPARE_DATE" USING

BY REFERENCE DATE_DEB_UTIL, DT_SYS, ETAT1

CALL "COMPARE_DATE" USING

BY REFERENCE DT_SYS, DATE_FIN_UTIL, ETAT2

MOVE "" TO ETAT3

IF ETAT1 = "E3A"

CALL "COMPARE_HEURE" USING

BY REFERENCE HEURE_DEB_UTIL, HEURE, ETAT3

END-IF

IF ETAT1 NOT EQUAL "SUP" AND ETAT2 NOT EQUAL "SUP"

AND ETAT3 NOT EQUAL "SUP"

MOVE 0 TO STOPPE MOVE 0 TO Z MOVE 1 TO EXISTE

IF HV>1

PERFORM CONTROL_VAL_OK UNTIL STOPPE = 1

END-IF

IF HV=1

PERFORM COPIE_MEMBRE VARYING Z FROM 1 BY 1

UNTIL STOPPE=1

END-IF

IF EXISTE = 1 MOVE 1 TO FINI

END-IF

FETCH LAST WITHIN KEEP_LIST

FREE LAST WITHIN KEEP_LIST

ELSE FETCH LAST WITHIN KEEP_LIST

FREE LAST WITHIN KEEP_LIST

ELSE FETCH LAST WITHIN KEEP_LIST

FREE LAST WITHIN KEEP_LIST

END-IF.

*calcul
d'
actualite
de la
valeur*

*contrôle
d'actualite*

*** La valeur est-elle
liée aux autres
valeurs comparées?*

*A LA SORTIE DE
CONTROL_VAL, si EXISTE
ne vaut pas 1 alors c'est que
la valeur trouvée est
totalement bonne*

** si le critère n'est pas
plus long que 1 alors
on peut passer à la
recherche dans TABLEAU_VALS
sans faire d'autres
vérifications*

*

CONTROL_VAL_OK.

** Cette primitive accede a l'article LIEN_REL suivant dans le chemin LIEN_RBC
** S'il existe et que l'item REL_COMP_LR (sa moitie de gauche) reference un
** bareme du critere de recherche, on controle si la moitie de droite egale l.
** partie de droite d'un des champs 'REL. Dans ce cas, l'article REL_BAR_CODE
** trouve est toujours presume valable, sinon il est a rejeter.

```
FETCH NEXT LIEN_REL WITHIN LIEN_RBC
  AT END MOVE 1 TO STOPPE.
IF STOPPE = 0
  MOVE REL_COMP_LR TO ELT
  COMPUTE Z = Z + 1
  MOVE IDV TO ELT2(1,Z)
  PERFORM VERIF_MEMBRE VARYING J FROM 2 BY 1 UNTIL J>NV
                                OR EXISTE=0
  IF EXISTE = 0 MOVE 1 TO STOPPE
  END-IF
END-IF.
```

VERIF_MEMBRE.

** Sous procedure de CONTROL_VAL_OK qui s'occupe du controle mentionne
** a la primitive precedente.

```
MOVE MREL(J) TO TRUC.
IF REL_COMP_LR NOT EQUAL MREL(1)
  IF (IDB=IDBT) AND (IDV NOT EQUAL IDVT)
    MOVE 0 TO EXISTE
  END-IF
  IF (IDB=IDBT) AND (IDV=IDVT)
    COMPUTE J = NV + 1
  END-IF
ELSE COMPUTE J = NV + 1.
```

COPIE_MEMBRE.

** Cette primitive accede aux nombres LIEN_REL suivant du chemin LIEN_RBC et
** recopie la partie de droite de l'item REL_COMP_LR dans TABLEAU_VALS.

```
FETCH NEXT LIEN_REL WITHIN LIEN_RBC
  AT END MOVE 1 TO STOPPE.
IF STOPPE = 0
  MOVE REL_COMP_LR TO ELT
  MOVE IDV TO ELT2(1,Z)
  END-IF.
```

*

TETE_BAR.

** Cette primitive accede aux records BAREME, LIEN_BAR et LIB_BAR du bareme
** dont le numero est ID_BAR et memorise les items trouves dans les arguments
** appropries.

```
MOVE ID_BAR TO NR_BAR.  
MOVE 1 TO EXISTE.  
FETCH NEXT BAREME USING NR_BAR  
    AT END MOVE 0 TO EXISTE.  
IF EXISTE = 1  
    MOVE DATE_CREATION OF BAREME TO DT_CR  
    MOVE TYPE_VAL TO TP_VAL  
    MOVE LONG_VAL TO LG_VAL  
    MOVE NBR_VERS_GARD TO NB_VERS  
    MOVE DUR_RETENTION TO DR_RET  
    MOVE 0 TO FINI  
    PERFORM EXTR_LIB_CD VARYING I FROM 1 BY 1 UNTIL FINI=1  
    COMPUTE NL = I - 2  
    MOVE 0 TO FINI  
    PERFORM EXTR LIEN_BAR VARYING I FROM 1 BY 1 UNTIL FINI=1  
    COMPUTE NB = I - 2.
```


NIVEAU4 SECTION.

BOUCLE_ACCES.

** Cette primitive trouve dans le bareme de numero ID_BAR, la valeur suivante
** qui n'est plus d'actualite.

```
MOVE ID_BAR TO NRBAR_REL.  
FETCH NEXT REL_BAR_CODE USING NRBAR_REL  
    AT END MOVE 1 TO FINI MOVE 1 TO STOPPE.  
IF FINI=0  
    CALL "TIMER" USING BY REFERENCE DT_SYS, HEURE  
    CALL "COMPARE_DATE" USING BY REFERENCE DATE_FIN_UTIL, DT_SYS, ETAT1  
    IF ETAT1="INH" MOVE 1 TO STOPPE.
```

*

TROUVE_VAL_SEO.

** Cette primitive accede a l'article REL_BAR_CODE suivant dont le numero
** de bareme est NRBAR_REL. S'il existe, on exerce dessus un controle d'actua-
** lité qui, s'il réussit, provoque la memorisation des valeurs adequates dans
** TABLEAU_VALS. Ensuite si le tableau est plein (8 lignes), STOPPE doit etre
** mis a 1 et HLIC a 8. Si ce REL_BAR_CODE suivant n'existe pas, STOPPE doit
** valoir 1 et HLIC doit valoir J-1.

FETCH NEXT REL_BAR_CODE USING NRBAR_REL
AT END MOVE 1 TO TERMI

COMPUTE HLIC = J - 1
MOVE 1 TO STOPPE.

IF STOPPE = 0

CALL "TIMER" USING BY REFERENCE DT_SYS, HEURE

CALL "COMPARE_DATE" USING BY REFERENCE

DATE_DEB_UTIL, DT_SYS, ETAT1

CALL "COMPARE_DATE" USING BY REFERENCE

DT_SYS, DATE_FIN_UTIL, ETAT2

MOVE "" TO ETAT3

IF ETAT1="EGA" CALL "COMPARE_HEURE" USING BY REFERENCE
HEURE_DEB_UTIL, HEURE, ETAT3

END-IF

IF ETAT1 NOT EQUAL "SUP" AND ETAT2 NOT EQUAL "SUP"

AND ETAT3 NOT EQUAL "SUP"

PERFORM REMPLIS_TABLEAU_VALS

END-IF

IF ETAT1="SUP" OR ETAT2="SUP" OR ETAT3="SUP" SUBTRACT 1 FROM J

END-IF

END-IF.

IF J=00 MOVE 1 TO STOPPE

MOVE 08 TO HLIC.

TROUVE_IDBAR.

** Cette primitive doit acceder a l'article BAREME suivant pour en memoriser
** le NR_BAR. S'il n'y en a plus, FINI doit valoir 1.

FETCH NEXT BAREME

AT END MOVE 1 TO FINI.

IF FINI=0 MOVE NR_BAR TO NBAR(I).

*

TROUVE_LB_SEQ.

** Cette primitive doit accéder à l'article LIEN_BAR suivant dont le numero
** de bareme est NR_BAR_LB et s'il existe doit remonter à l'owner BAREME du
** chemin EST_COMP_DE dont cet article est membre pour en memoriser le NR_BAR

```
IF I>1 FETCH      LAST WITHIN KEEP_LIST
      FREE      LAST WITHIN KEEP_LIST.
FIND NEXT LIEN_BAR USING NR_BAR_LB
      AT END MOVE 1 TO FINI.
IF FINI=0 KEEP CURRENT USING KEEP_LIST
      FETCH OWNER WITHIN EST_COMP_DE
      MOVE NR_BAR TO NBAR(I).
```

TRSF_LIB_CD.

** Cette primitive doit memoriser un article LIB_BAR dans la base de
** donnees apres y avoir rempli les items.

```
MOVE LIBELLE (I) TO VAL_LIB_BAR.
MOVE REC_LING(I) TO CODE_RL_BAR.
STORE LIB_BAR.
```

TRSF_LIB_VAL.

** Cette primitive doit memoriser un article LIB_VAL_CODE dans la base de
** donnees apres y avoir rempli les items.

```
MOVE LIBELLE (I) TO VAL_LIB_LVT.
MOVE REC_LING(I) TO CODE_RL_LVT.
STORE LIB_VAL_CODE.
```

TRSF LIEN_BAR.

** Cette primitive doit memoriser un article LIEN_BAR dans la base de
** donnees apres y avoir rempli les items.

```
MOVE IDENT_BAR(I) TO NR_BAR_LB.
MOVE ROLE_BAR(I) TO ROLE_LB.
STORE LIEN_BAR.
```

TRSF LIEN_REL. ** Memorise un article LIEN_REL apres y avoir rempli les item

```
MOVE NREL(I) TO REL_COMP_LR.
MOVE PREL(I) TO ROLE_LR.
STORE LIEN_REL.
```

*

CONTROL_PERINE.

** Cette primitive supprime la valeur de bareme\$qu'on lui fournit si sa
** duree de retention est ecoulee.

```
CALL "CALCUL_DATE" USING BY REFERENCE DATE_FIN_UTIL,DR_RET,DT_PE.  
CALL "TIMER" USING BY REFERENCE DT_SYS,HEURE.  
CALL "COMPARE_DATE" USING BY REFERENCE DT_SYS,DT_PE,ETAT1.  
IF ETAT1="SUP"  
    DISPLAY "Je supprime la valeur      : ",NRBAR_VALCODE  
  
    FRASE REL_BAR_CODE  
END-IF.
```

REEMPLIS_TABLEAU_VALS.

** Cette primitive doit remplir la ligne J de TABLEAU_VALS par les valeurs
** appropriees de l'article REL_BAR_CODE trouve et des articles LIB_VAL_CODE
** qui y sont lies.

```
MOVE NRBAR_VALCODE TO IDBV.  
MOVE IDV TO ELT2(J,11).  
IF NB>0 PERFORM EXTR_VALS_LR VARYING Z FROM 1 BY 1 UNTIL Z>NB.
```

EXTR_VALS_LR.

** Cette primitive accede a l'article LIEN_REL suivant dans le chemin LIEN_RBC
** et extrait la partie de droite de l'item REL_COMP_LR pour la memoriser dan
** TABLEAU_VALS.

```
FETCH NEXT LIEN_REL WITHIN LIEN_RBC.  
IF Z<NB OR Z=NB MOVE REL_COMP_LR TO IDBV  
    MOVE IDV TO ELT2(J,Z).
```

EXTR_LIB_CD.

** Cette primitive accede a l'article LIB_BAR suivant dans le chemin
** LIEN_BAR_LIB et memorise ses valeurs d'item\$ dans LIBS_BAR si FINI vaut 0.
** S'il n'existe pas, FINI doit valoir 1.

```
FETCH NEXT LIB_BAR WITHIN LIEN_BAR_LIB  
    AT END MOVE 1 TO FINI.  
IF FINI=0 MOVE VAL_LIB_BAR TO LIBELLE(I)  
    MOVE CODE_LIB_BAR TO REG-LING(I).
```

*

EXTR-LIEN-BAR.

** Cette primitive accede a l'article LIEN-BAR suivant dans le chemin
** EST_COMP_DE et memorise ses valeurs d'items dans TAB-BARS si FINI vaut 0.
** S'il n'existe pas, FINI doit valoir 1.

FETCH NEXT LIEN-BAR WITHIN EST_COMP_DE
AT END MOVE 1 TO FINI.
IF FINI=0 MOVE NR-BAR-LB TO IDENT-BAR(I)
MOVE ROLE-LB TO ROLE-BAR(I).

EXTR-LIB-VAL.

** Cette primitive accede a l'article LIB-VAL_CODE suivant dans le chemin
** LIEN_REL_LIB et memorise ses valeurs d'items si FINI vaut 0.
** S'il n'existe pas, FINI doit valoir 1.

FETCH NEXT LIB-VAL_CODE WITHIN LIEN_REL_LIB
AT END MOVE 1 TO FINI.
IF FINI=0 MOVE VAL-LIB-LVT TO LIBELLE(I)
MOVE CODE-RL-LVT TO REG-LING(I).

EXTR-LIEN-REL.

** Cette primitive accede a l'article LIEN-REL suivant dans le chemin
** LIEN_RPC et memorise ses valeurs d'items si FINI vaut 0.
** S'il n'existe pas, FINI doit valoir 1.

FETCH NEXT LIEN-REL WITHIN LIEN_RPC
AT END MOVE 1 TO FINI.
IF FINI=0 MOVE REL_COMP_LR TO NREL(I)
MOVE ROLE_LR TO RREL(I).

EXTR-LIEN-BAR.

** Cette primitive accede a l'article LIEN-BAR suivant dans le chemin
** EST_COMP_DE et memorise ses valeurs d'items dans TAB-BARS si FINI vaut 0.
** S'il n'existe pas, FINI doit valoir 1.

FETCH NEXT LIEN-BAR WITHIN EST_COMP_DE
AT END MOVE 1 TO FINI.
IF FINI=0 MOVE NR-BAR-LB TO IDENT-BAR(I)
MOVE ROLE-LB TO ROLE-BAR(I).

EXTR-LIB-VAL.

** Cette primitive accede a l'article LIB-VAL_CODE suivant dans le chemin
** LIEN_REL-LIB et memorise ses valeurs d'items si FINI vaut 0.
** S'il n'existe pas, FINI doit valoir 1.

FETCH NEXT LIB-VAL_CODE WITHIN LIEN_REL-LIB
AT END MOVE 1 TO FINI.
IF FINI=0 MOVE VAL-LIB-LVT TO LIBELLE(I)
MOVE CODE-RL-LVT TO REG-LING(I).

EXTR-LIEN-REL.

** Cette primitive accede a l'article LIEN-REL suivant dans le chemin
** LIEN_RPC et memorise ses valeurs d'items si FINI vaut 0.
** S'il n'existe pas, FINI doit valoir 1.

FETCH NEXT LIEN-REL WITHIN LIEN_RPC
AT END MOVE 1 TO FINI.
IF FINI=0 MOVE REL_COMP-LR TO NREL(I)
MOVE ROLE-LR TO RREL(I).

Programme ES.COB

IDENTIFICATION DIVISION.

PROGRAM-ID. MODULE-ENTREE-SORTIE.

AUTHOR. E EVARD.

DATE-WRITTEN. 16/04/1986.

**
** MODULE ENTREE - SORTIE **
**

** Ce module a pour but de repondre a la majorite des demandes en entrees/
** sorties interactives du projet de gestion interactive de baremes.

ENVIRONNEMENT DIVISION.

CONFIGURATION SECTION.

SOURCE-COMPUTER. vax_11.
OBJECT-COMPUTER. vax_11.

DATA DIVISION.

WORKING-STORAGE SECTION.

01 FUNCTION PIC S9(9) USAGE IS COMP VALUE IS 122.
01 STAT PIC S9(9) USAGE IS COMP VALUE IS 0.
 88 SSS-NORMAL; VALUE IS 1.
01 CHANNEL PIC S9(9) USAGE IS COMP.
01 TT PIC X(3); VALUE IS "TT".
01 LIGNE-TITRE1.
 02 FILLER; PIC X(3);VALUE "CODE".
 02 PAP1; PIC X(12).
 02 FILLER; PIC X(2);VALUE " ".
 02 PAP2; PIC X(12).
 02 FILLER; PIC X(2);VALUE " ".
 02 PAP3; PIC X(12).
 02 FILLER; PIC X(2);VALUE " ".
 02 PAP4; PIC X(12).
 02 FILLER; PIC X(2);VALUE " ".
 02 PAP5; PIC X(12).
01 LIGNE-TITRE2.
 02 PAP6; PIC X(12).
 02 FILLER; PIC X(2);VALUE " ".
 02 PAP7; PIC X(12).
 02 FILLER; PIC X(2);VALUE " ".
 02 PAP8; PIC X(12).


```

02 FILLER; PIC X(2);VALUE " ".
02 PAR9; PIC X(12).
02 FILLER; PIC X(2);VALUE " ".
02 PAR10; PIC X(12).
01 TIRET1.
02 FILLER;PIC X(8);VALUE "-----".
01 TIRET2.
02 FILLER;PIC X(14);VALUE "-----".
01 LIGNE_VALEURS1.
02 VAL0 PIC X(6).
02 FILLER PIC X(2).
02 VAL1 PIC X(6).
02 FILLER PIC X(8).
02 VAL2 PIC X(6).
02 FILLER PIC X(8).
02 VAL3 PIC X(6).
02 FILLER PIC X(8).
02 VAL4 PIC X(6).
02 FILLER PIC X(8).
02 VAL5 PIC X(6).
01 LIGNE_VALEURS2.
02 VAL6 PIC X(6).
02 FILLER PIC X(8).
02 VAL7 PIC X(6).
02 FILLER PIC X(8).
02 VAL8 PIC X(6).
02 FILLER PIC X(8).
02 VAL9 PIC X(6).
02 FILLER PIC X(8).
02 VAL10 PIC X(6).
01 LIGNE_TIRETS.
02 FILLER;PIC X(40);VALUE"-----".
02 FILLER;PIC X(36);VALUE"-----".
01 HEURE PIC X(5).
01 ETAT PIC X(3).
01 RESTE PIC 9(2).
01 RESUL PIC 9(2).
01 HOUR_FIELD.
02 CHP_HR PIC X(2).
02 FILLER;PIC X(1);VALUE IS "H".
02 CHP_MM PIC X(2).
01 TIME_FIELD PIC X(10).
01 DATE_FIELD.
02 CHP_JR PIC X(2).
02 FILLER;PIC X(1);VALUE IS "/".
02 CHP_MS PIC X(2).
02 FILLER;PIC X(1);VALUE IS "/".
02 CHP_AN PIC X(4).
02 CHP_AN_NUM;REDEFINES CHP_AN;PIC 9(4).
01 J PIC 9(2).
01 BIDON PIC X(1).
01 STOPPE PIC 9(1).
01 Z PIC 9(2).
01 H PIC 9(2).
01 NUM_UTIL.
02 NUM_UTIL PIC X(10).
02 NUM_PASS PIC X(10).
02 "REP" ARITHMETIC UTILPASS; PIC X; OCCURS 10 TIMES .

```

LINKAGE SECTION.

*-----

```

01 ID_UTIL PIC X(20).
01 IDH_PROG PIC 9(2).
01 IDH_MESS PIC 9(2).
01 EXISTE PIC 9(1).
01 ID_BAR PIC X(3).
01 IL PIC 9(2).
01 LIBS_BAR.
    02 ELEMENT OCCURS 1 TO 10 TIMES DEPENDING ON NL.
        03 LIPHEF PIC X(50).
        03 REC_LING PIC X(2).
01 DT_CR PIC X(10).
01 TP_VAL PIC X(2).
01 LG_VAL PIC X(2).
01 IB_VERS PIC X(2).
01 DR_RET PIC X(4).
01 IB PIC 9(2).
01 IBP PIC 9(2).
01 TAB_BARS.
    02 ELEMENT OCCURS 1 TO 10 TIMES DEPENDING ON NB.
        03 IDENT_BAR PIC X(6).
        03 POLE_BAR PIC X(50).
        03 DJT PIC 9(4).
01 I PIC 9(2).
01 LIBLU PIC X(50).
01 IV PIC 9(2).
01 TAB_RELS.
    02 ELEMENT OCCURS 20 TIMES.
        03 MREL PIC X(12).
        03 PREL PIC X(50).
01 FINI PIC 9(1).
01 OPT_MENU PIC 9(2).
01 DT_DB PIC X(10).
01 HR_DB PIC X(5).
01 DT_FI PIC X(10).
01 VAL_CODE PIC X(6).
01 ID_REL PIC X(6).
01 REP PIC X(2).
01 TABLEAU_VALS.
    02 ELT1 OCCURS 8 TIMES.
        03 ELT2 PIC X(6); OCCURS 11 TIMES.
01 NLIC PIC 9(2).
01 TABLEAU_LIPS.
    02 LIPS; PIC X(12); OCCURS 10 TIMES.
01 LIBBAR PIC X(50).
*
```



```

*****
PROCEDURE DIVISION USING NUM_PROC, NUM_MESS, ID_BAR, LIBS_BAR, NL,
                        TAB_BARS, HS, HBP, DT_CR, TP_VAL, LG_VAL,
                        NB_VERS, DR_RET, LIBAN, EXISTE, FINI, I,
                        OPT_MENU, OC_DB, HR_DB, DT_FN, VAL_CODE,
                        ID_REL, NV, TAB_RELS, REP, TABLEAU_VALS, ILIG,
                        TABLEAU_LIBS, LIBBAR, ID_UTIL.
*****

```

```

*****
NIVEAU1 SECTION.
*****

```

```

*****
SUECTION.

```

** Voici le programme general du module. Il a pour but de determiner la
 ** procedure a executer selon la valeur de la variable NUM_PROC.

```

IF NUM_PROC = 00. PERFORM ALL_MESSAGE.
IF NUM_PROC = 01. PERFORM CLEAR_SCREEN.
IF NUM_PROC > 01 AND NUM_PROC < 27 MOVE 99 TO NUM_MESS.
IF NUM_PROC = 02. PERFORM LECT_ID_BAR.
IF NUM_PROC = 03. PERFORM LECT_HBP.
IF NUM_PROC = 04. PERFORM LECT_NL.
IF NUM_PROC = 05. PERFORM LECT_BAR_REMS.
IF NUM_PROC = 06. PERFORM LECT_BAR_COMP.
IF NUM_PROC = 07. PERFORM LECT_LG_VAL.
IF NUM_PROC = 08. PERFORM LECT_NB_VERS.
IF NUM_PROC = 09. PERFORM LECT_DR_RET.
IF NUM_PROC = 10. PERFORM LECT_LIB.
IF NUM_PROC = 11. PERFORM SORT_LIB.
IF NUM_PROC = 12. PERFORM SORT_IDBAR.
IF NUM_PROC = 13. PERFORM LECT_BIBLI.
IF NUM_PROC = 14. PERFORM SORTIR_BAR.
IF NUM_PROC = 15. PERFORM LECT_OPT_MENU.
IF NUM_PROC = 16. PERFORM LECT_ROLE_BAR.
IF NUM_PROC = 17. PERFORM LECT_REL_REMS_LIBS.
IF NUM_PROC = 18. PERFORM LECT_VAL_CODE.
IF NUM_PROC = 19. PERFORM LECT_ROLE_REL.
IF NUM_PROC = 20. PERFORM SORTIR_REL.
IF NUM_PROC = 21. PERFORM LECT_REP.
IF NUM_PROC = 22. PERFORM LECT_DATE_HEURE.
IF NUM_PROC = 23. PERFORM SORTIR_DATE_HEURE.
IF NUM_PROC = 24. PERFORM SORTIR_TABLEAU1.
IF NUM_PROC = 25. PERFORM SORTIR_TABLEAU2.
IF NUM_PROC = 26. PERFORM IDENTIFIER.
IF NUM_PROC = 27. PERFORM LECT_CHOIX.

```

EXIT PROGRAM.

```

*****
*

```

NIVEAU2 SECTION.

CLEAR_SCREEN. (n°1)

** Cette primitive efface l'écran.

Postcond: écran effacé.

CALL "SCSRASH_PAGE" USING BY VALUE 1,1.

IDENTIFIER. (n°26)

** Cette primitive s'occupe de lire à l'écran l'identification d'un utilisateur. La lecture du mot de passe se fait sans echo au terminal.

MOVE "" TO ID_UTIL.

PERFORM GET_UTIL_UTIL UNTIL ID_UTIL NOT EQUAL "".

PERFORM M1573.

CALL "SYSSASSIGN" USING BY DESCRIPTOR TT,
BY REFERENCE CHANNEL,
BY VALUE 0,
BY VALUE 0,
GIVING STAT.

argument: ID_UTIL

postcond: ID_UTIL non vide
et contenant dans sa
partie gauche le nom
partie droite le mot de
passe.

IF STAT IS FAILURE DISPLAY "SYSTEM_STATUS=", STAT.

MOVE 0 TO FINI.

PERFORM ACC_NUMCHO VARYING Z FROM 1 BY 1 UNTIL Z>10 OR FINI=1.

CALL "SYSSDASSG1" USING BY VALUE CHANNEL.

IF STAT IS FAILURE DISPLAY "SYSTEM_STATUS=", STAT.

MOVE ID_UTIL TO ID_UTIL.

EXIT PROGRAM.

ACC_NUMCHO.

** Cette primitive procede a la lecture sans echo de la variable BIDDON.

CALL "SYSSQIDON" USING BY VALUE 0,
BY VALUE CHANNEL,
BY VALUE FUNCTION,
BY VALUE 0,
BY VALUE 0,
BY VALUE 0,
BY REFERENCE BIDDON,
BY VALUE 1,
BY VALUE 0,
BY VALUE 0,
BY VALUE 0,
BY VALUE 0,

GIVING STAT.

IF STAT IS FAILURE DISPLAY "SYSTEM_STATUS=", STAT.

IF BIDDON<" " MOVE 1 TO FINI ELSE MOVE BIDDON TO ID_UTIL.

LECT_IOM_UTIL.

** Cette primitive lit un nom d'utilisateur.

PERFORM M372.
ACCEPT IOM_UTIL.

LECT_REP. (n°21)

** Cette primitive lit une réponse à l'écran.

ACCEPT REP.

arg: REP
précond: /
postcond: /

LECT_CHOIX. (n°24)

** Cette primitive lit à l'écran la variable REP jusqu'à ce qu'elle soit
** égale à "O" ou "N".

MOVE " " TO REP.
PERFORM LECT_BOUCLE_CHOIX UNTIL REP="O" OR REP="N".

arg: REP
postcond: REP ∈ {"O", "N"}

LECT_BOUCLE_CHOIX.

** Corps de la boucle de la primitive précédente.

PERFORM CLEAR_SCREEN.
PERFORM M379.
PERFORM LECT_REP.

LECT_ROLE_BAR. (n°16)

** Cette primitive lit à l'écran le champ ROLE_BAR de la composante numero I
** dans le tableau TAB_BARS.

PERFORM M320.
ACCEPT ROLE_BAR(I).

arg: I, ROLE_BAR(I). précond: 01 ≤ I ≤ 10.

LECT_ROLE_RRL. (n°19)

** Cette primitive lit à l'écran le champ RREL de la composante numero I
** dans le tableau TAB_RRLS.

PERFORM M345.
ACCEPT RREL(I).

arg: I, RREL(I). précond: 01 ≤ I ≤ 10.

*

LECT_OPT_MENU. (n°15)

** Cette primitive lit a l'ecran une option de menu.

ACCEPT OPT_MENU.

arg: OPT_MENU

precond: /

postcond: /

LECT_ID_BAR. (n°2)

** Cette primitive lit a l'ecran un identificateur de bareme tant que celui-ci ne contient pas de valeur.

MOVE 0 TO STOPPE.

PERFORM BOUCLE_ID_BAR UNTIL STOPPE=1.

arg: ID_BAR

postcond: ID_BAR non vide

BOUCLE_ID_BAR.

** Corps de la boucle de la procedure LECT_ID_BAR.
** Si ID_BAR est valide, alors STOPPE est mis a 1.

PERFORM MSG06.

ACCEPT ID_BAR.

IF ID_BAR NOT EQUAL "" MOVE 1 TO STOPPE

ELSE DISPLAY "" PERFORM MSG50.

LECT_VAL_CODE. (n°18)

** Cette primitive lit une valeur de code a l'ecran.

ACCEPT VAL_CODE.

arg: VAL_CODE.

precond: /

post: /

LECT_NBP. (n°3)

** cette primitive lit la variable NBP et controle si elle est numerique et comprise entre 1 et 10. Si elle est valide, FINI est mis a 1.

PERFORM MSG07.

ACCEPT NBP.

IF NBP NOT NUMERIC OR NBP < 00 OR NBP > 10

PERFORM MSG29

ELSE MOVE 1 TO FINI.

arg: NBP.

postcond: 01 \$NBP < 10
numerique

*

LECT_NL. (n°4)

** Cette primitive lit la variable NL, controle si elle est numerique et
** comprise entre 01 et 10. Si elle est valide, alors FINI est mis a 1.

PERFORM MCS02.
ACCEPT NL.
IF NL NOT NUMERIC OR NL < 01 OR NL > 10.
 PERFORM MCS20
ELSE MOVE 1 TO FINI.

arg: NL, FINI

postcond: { NL numerique
 1 ≤ NL ≤ 10

LECT_BIDON. (n°43)

ACCEPT BIDON.

precond: ✓
arg: BIDON

postcond: ✓

LECT_BAR_PEMS. (n°5)

** Cette primitive lit les renseignements generaux d'un bareme excepte
** ses libelles.

PERFORM MCS61.
PERFORM MCS02.
CALL "TIMER" USING BY REFERENCE TIME_FIELD, HEURE.
MOVE TIME_FIELD TO DT_CR.DISPLAY DT_CR.
MOVE 0 TO FINI.
PERFORM DECT_TP_VAL UNTIL FINI=1.
MOVE 0 TO FINI.
PERFORM DECT_LG_VAL UNTIL FINI=1.
MOVE 0 TO FINI.
PERFORM DECT_NB_VERS UNTIL FINI=1.
MOVE 0 TO FINI.
PERFORM DECT_NL UNTIL FINI=1.

arg: DT_CR, TP_VAL, LG_VAL, NB_VERS, NL

postcond: { DT_CR = date système
 TP_VAL ∈ { "AN", "NU", "AB" }
 01 ≤ LG_VAL ≤ 06
 00 ≤ NB_VERS ≤ 20
 01 ≤ NL ≤ 10

LECT_TP_VAL.

** Cette primitive doit lire la variable TP_VAL et controler si elle vaut
** "AB", "AN", ou "NU". Si c'est le cas, FINI est mis a 1 sinon 0.

PERFORM MCS10.
ACCEPT TP_VAL.
IF TP_VAL = "AB"
 OR TP_VAL = "AN"
 OR TP_VAL = "NU"
 MOVE 1 TO FINI
ELSE
 DISPLAY "Possibilites"
 DISPLAY "
 DISPLAY "

: AB = AlphaBetique"
 AN = AlphaNumerique"
 NU = Numerique".

*

LECT-NEE-PEMB-LIBS. (n°17)

** Cette primitive lit et valide la variable NL et ensuite procede a la
** lecture de NL libelle(s).

```
MOVE 0 TO FINI.  
PERFORM LECT_NL UNTIL FINI=1.  
PERFORM MES62.  
PERFORM LECT_LIBS VARYING I FROM 1 BY 1 UNTIL I > NL.
```

arg: NL, LIBS-BAR.

postcond: $01 \leq NL \leq 10$ et numérique
LIBS-BAR contient NL
entrées non vides

LECT-DATE-HEURE. (n°22)

** Cette primitive doit lire et valider les dates et heures qui concernent
** une valeur de code.

```
PERFORM MES69.  
CALL "TIMER" USING BY REFERENCE TIME-FIELD, HEURE.  
MOVE TIME-FIELD TO DT-CR. DISPLAY DT-CR.  
MOVE 0 TO STOPPE.  
PERFORM LECT_DT_DB UNTIL STOPPE=1.  
MOVE 0 TO STOPPE.  
PERFORM LECT_HEURE UNTIL STOPPE=1.  
MOVE HOUR-FIELD TO HR-DB.  
MOVE 0 TO STOPPE.  
PERFORM LECT_DT_FN UNTIL STOPPE=1.
```

rem: { Il est plus facile d'effectuer les validations sémantiques (cfr page)
en entrée pour ces dates et cette heure.

arg: DT-CR, DT-DB, HR-DB, DT-FN.

postcond: DT-CR, DT-DB, DT-FN sont
des dates valides % calendriers
- HR-DB est une heure valide.
 $\geq DT-CR \leq DT-DB \leq DT-FN$
* si DT-CR = DT-DB alors HR-DB //
heure n'est pas

LECT-DT-FN.

** Cette primitive lit une date DT-FN et controle si elle est bien superieure
** ou egale a celle contenue dans DT-DB.

```
PERFORM MES69.  
PERFORM LECT-DATE.  
MOVE DATE-FIELD TO DT-FN.  
IF STOPPE=1  
CALL "COMPARE-DATE" USING BY REFERENCE DT-DB, DT-FN, ETAT  
IF ETAT = "SUP"  
MOVE 0 TO STOPPE  
PERFORM MES67  
END-IF  
END-IF.
```

*

LECT_DT_DB.

** Cette primitive lit une date DT_DB et controle si elle est bien superieure
** ou egale a DT_CR.

```
PERFORM MES37.  
PERFORM LECT_DATE.  
MOVE DATE_FIELD TO DT_DB.  
IF STOPPE=1  
  CALL "COMPARE_DATE" USING BY REFERENCE DT_CR,DT_DB,ETAT  
  IF ETAT = "SUP"  
    MOVE 0 TO STOPPE  
    PERFORM MES56  
  END-IF  
END-IF.
```

LECT_HEURE.

** Cette primitive lit une heure HOUR_FIELD (heure-minute), la valide par
** rapport a l'horloge. De plus, si DT_DB est egal a DT_CR, cette heure
** sera acceptee uniquement si elle est superieure ou egale a l'heure du
** systeme. Dans ce cas, STOPPE est mis a 1 sinon 0.

```
PERFORM MES38.  
MOVE 1 TO STOPPE.  
ACCEPT HOUR_FIELD.  
IF CHP_HR NOT NUMERIC OR CHP_HR<00 OR CHP_HR>23  
  MOVE 0 TO STOPPE  
  PERFORM MES51.  
IF CHP_MIN NOT NUMERIC OR CHP_MIN<00 OR CHP_MIN>59  
  MOVE 0 TO STOPPE  
  PERFORM MES52.  
CALL "TIMER" USING BY REFERENCE TIME_FIELD,HEURE.  
CALL "COMPARE_DATE" USING BY REFERENCE DT_DB,DT_CR,ETAT.  
IF ETAT="EGA"  
  CALL "COMPARE_HEURE" USING BY REFERENCE HOUR_FIELD,HEURE,ETAT  
  IF ETAT="INF"  
    MOVE 0 TO STOPPE  
    PERFORM MES33  
  END-IF  
END-IF.
```

*

LECT-DATE.

** Cette primitive a pour but de lire une date DATE_FIELD et de la valider
** par rapport au calendrier. Si elle est valide, STOPPE est mis a 1 sinon 0.

```
MOVE 1 TO STOPPE.
ACCEPT DATE_FIELD.
IF CHP_AN NOT NUMERIC OR CHP_JR NOT NUMERIC OR CHP_MS NOT NUMERIC
    PERFORM MES53
    MOVE 0 TO STOPPE.
IF CHP_MS < 01 OR CHP_MS > 12 OR CHP_AN < 00
    PERFORM MES54
    MOVE 0 TO STOPPE.
IF (CHP_MS=01 AND CHP_JR>31) OR (CHP_MS=03 AND CHP_JR>31) OR
   (CHP_MS=04 AND CHP_JR>30) OR (CHP_MS=05 AND CHP_JR>31) OR
   (CHP_MS=06 AND CHP_JR>30) OR (CHP_MS=07 AND CHP_JR>31) OR
   (CHP_MS=08 AND CHP_JR>31) OR (CHP_MS=09 AND CHP_JR>30) OR
   (CHP_MS=10 AND CHP_JR>31) OR (CHP_MS=11 AND CHP_JR>30) OR
   (CHP_MS=12 AND CHP_JR>31)
    PERFORM MES55 MOVE 0 TO STOPPE.
IF CHP_AN NUMERIC
    DIVIDE CHP_AN NUM BY 4 GIVING RESUL REMAINDER RESTE.
IF (CHP_MS=02 AND RESTE>00)
    IF CHP_JR>28 PERFORM MES55
    MOVE 0 TO STOPPE
END-IF.
END-IF.
IF (CHP_MS=02 AND RESTE=00)
    IF CHP_JR>28 PERFORM MES55
    MOVE 0 TO STOPPE
END-IF.
END-IF.
```

LECT-BAR-COMP. (n°06) ** Cette primitive lit le libelle d'un bareme.

```
PERFORM MES14.
ACCEPT LIBBJ.
```

{ arg: LIBLU
postcond: /

LECT-LG-VAL. (n°07)

** Cette primitive lit la variable LG_VAL (longueur maximum d'une valeur de
** code) et controle si elle est numerique et comprise entre 1 et 6. Si c'est
** le cas, FINI est mis a 1 sinon 0.

```
PERFORM MES11.
ACCEPT LG_VAL.
IF LG_VAL NOT NUMERIC OR LG_VAL < 01 OR LG_VAL > 06
    MOVE 0 TO FINI
ELSE MOVE 1 TO FINI.
```

{ arg: LG-VAL
postcond: $01 \leq LG_VAL \leq 06$
et numerique

#

LECT-NB-VERS. (n°9)

** Cette primitive lit la variable NB-VERS (nombre maximum de versions
 ** successives d'une valeur a garder) et controle si elle est numerique et
 ** comprise entre 01 et 20. Si c'est le cas, FINI est mis a 1 sinon 0.

PERFORM MES12.
 ACCEPT NB-VERS.
 IF NB-VERS NOT NUMERIC OR NB-VERS<00 OR NB-VERS>20
 MOVE 0 TO FINI
 ELSE MOVE 1 TO FINI.

arg: NB-VERS
 postcond: $000 \leq \text{NB-VERS} \leq 20$
 et numerique

LECT-DR-RET. (n°9)

** Cette primitive lit la variable DR-RET (duree de retention) et controle si
 ** elle est numerique et superieure a 0. Dans ce cas, FINI est mis a 1 sinon 0.

PERFORM MES13.
 ACCEPT DR-RET.
 IF DR-RET NOT NUMERIC OR DR-RET<0001 MOVE 0 TO FINI
 ELSE MOVE 1 TO FINI.

arg: DR-RET
 postcond: $0001 \leq \text{DR-RET} \leq 9999$
 et numerique

LECT-LIB. (n°10)

** Cette primitive doit lire un libelle du tableau LIBS-BAR et son regime
 ** linguistique appropriée.

MOVE 0 TO FINI.
 PERFORM BOUCLE-LIB UNTIL FINI=1.
 MOVE 0 TO FINI.
 PERFORM BOUCLE-RL UNTIL FINI=1.

precond: $1 \leq I \leq 10$
 arg: I, LIBS-BAR
 postcond: $\left\{ \begin{array}{l} \text{LIBELLE(I) non vide} \\ \text{REG.LING(I)} \in \{ "AN", "FR", "NR", "AL" \} \end{array} \right\}$

BOUCLE-RL.

** Cette primitive doit lire et valider LIBELLE(I). Si la lecture est correcte
 ** FINI est mis a 1. (Corp de boucle)

PERFORM MES24.
 ACCEPT REG-LING(I).
 IF REG-LING(I)="AN" OR REG-LING(I)="FR" OR REG-LING(I)="NR"
 OR REG-LING(I)="AL" MOVE 1 TO FINI
 ELSE PERFORM MES85.

 *

BOUCLE_LIB.

** Cette primitive doit lire et valider RES_LING(I). Si la lecture est correcte
** FINI est mis à 1. (Corp de boucle)

```
PERFORM MES23.  
ACCEPT LIBELLE(I).  
IF LIBELLE(I) NOT EQUAL "" MOVE 1 TO FINI  
ELSE PERFORM MES20.
```

SORT_LIB. ** Cette primitive sort le libelle LIBELLE(I) à l'écran.

```
PERFORM MES23. (Corp de boucle)  
DISPLAY LIBELLE(I).
```

SORT_IDBAR.(n°12)

** Cette primitive sort à l'écran l'identificateur d'un bareme composant et
** son role dans la composition. (Corp de boucle)

```
PERFORM MES25.  
DISPLAY IDENT-BAR(I).  
PERFORM MES26.  
DISPLAY ROLE-BAR(I).
```

arg: IDENT-BAR(I), ROLE-BAR(I), I
Precond: 1 ≤ I ≤ 10 et 2 autres non vides.
Postcond: 2 premiers arguments affichés à l'écran.

SORTIR_BAR. (n°14) ** Cette primitive sort un en-tete de bareme à l'écran.

```
IF EXISTE = 0 PERFORM MES21  
ELSE IF IB > 0 PERFORM MES28  
END-IF  
IF NB = 0 PERFORM MES27  
END-IF  
MOVE ID_UTIL TO NUM_UTIL  
DISPLAY "Son createur est ", NUM_UTIL, ". "  
DISPLAY ""  
PERFORM SORT_LIB VARYING I FROM 1 BY 1 UNTIL I > NL  
PERFORM MES06 DISPLAY ID_BAR  
PERFORM MES09 DISPLAY DT_CR  
PERFORM MES10  
IF TP_VAL = "AN" DISPLAY "Alphanumerique" END-IF  
IF TP_VAL = "AB" DISPLAY "Alphabetique" END-IF  
IF TP_VAL = "NU" DISPLAY "Numerique" END-IF  
PERFORM MES11 DISPLAY LG_VAL  
PERFORM MES12 DISPLAY IB_VERS  
PERFORM MES13 DISPLAY DR_RET.
```

arg: ID_UTIL, LIBS_BAR, NL, ID_BAR,
TP_VAL, LG_VAL, NB_VERS, DR_RET,
TAB_BARS, NB.

Precond: { arg. supposés syntaxiquement et
sémantiquement corrects.

Postcond: { le tout est affiché sauf TAB_BARS pour
le constant

```
(IF IB > 0 PERFORM SORT_IDBAR VARYING I FROM 1  
BY 1 UNTIL I > NB.
```

(se trouve ici en commentaires, on peut éventuellement
le supprimer!)

**
**

SORTIR_REL. (n° 20)

** Cette primitive sort une valeur de code a l'ecran associee a ses
** renseignements generaux.

```
IF EXISTE = 0 PERFORM MES40
ELSE IF NV > 0 PERFORM MES41
END-IF
IF NV = 0 PERFORM MES42
END-IF
DISPLAY ""
PERFORM MES43 DISPLAY VAL_CODE
DISPLAY ""
PERFORM SORT_LIB VARYING I FROM 1 BY 1 UNTIL I > NL
DISPLAY ""
PERFORM MES09 DISPLAY DT_CR
PERFORM MES37 DISPLAY DT_DB
PERFORM MES38 DISPLAY HR_DB
PERFORM MES39 DISPLAY DT_FN.
```

arg: VAL_CODE, LIBS_BAR, NL, DT_CR, DT_DB,
HR_DB, DT_FN

prend: { arguments tous supprimés sémantiquement
et syntactiquement corrects

produit: Tout est affiché à l'écran.

SORTIR_DATE_HEURE.

** Cette primitive sort a l'ecran les dates et l'heure comprises dans DT_CR,
** DT_DB, HR_DB et DT_FN.

```
PERFORM MES09.DISPLAY DT_CR.
PERFORM MES37.DISPLAY DT_DB.
PERFORM MES38.DISPLAY HR_DB.
PERFORM MES39.DISPLAY DT_FN.
DISPLAY "".
EXIT PROGRAM.
```

SORT_TIRET.

** sert au dessin du tableau de bareme a l'ecran.

```
DISPLAY TIRET2 WITH NO ADVANCING.
```

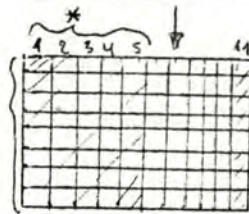
*

SORTIR_TABLEAU1. (N° 24)

** Cette primitive sort la partie de gauche d'un tableau de valeurs d'un
 ** bareme a l'ecran.
 ** Le titre du bareme est dans LIBBAR.
 ** Les titres des colonnes se trouvent dans TABLEAU_LIBS (de longueur NB).
 ** Les valeurs se trouvent dans TABLEAU_VALS (de longueur NLIG).

```

PERFORM CLEAR_SCREEN.
MOVE LIBS(1) TO BAR1.
MOVE LIBS(2) TO BAR2.
MOVE LIBS(3) TO BAR3.
MOVE LIBS(4) TO BAR4.
MOVE LIBS(5) TO BAR5.
DISPLAY LIGNE_TITRES.
DISPLAY "
DISPLAY LIGNE_TITRES.
DISPLAY LIGNE_TITRES.
DISPLAY LIGNE_TITRES.
DISPLAY TITRE1 WITH NO ADVANCING.
IF NB>5 PERFORM SORT_TITRE VARYING I FROM 1 BY 1 UNTIL I>5.
IF NB<6 PERFORM SORT_TITRE VARYING I FROM 1 BY 1 UNTIL I>NB.
DISPLAY ".
IF NLIG<03 MOVE NLIG TO 0 ELSE MOVE 06 TO NLIG.
PERFORM SORTIR_LIGNES_VALS VARYING Z FROM 1 BY 1 UNTIL Z>NLIG.
DISPLAY LIGNE_TITRES.
EXIT PROGRAM.
  
```



arg: TABLEAU_VALS, NLIG, LIBBAR,
 TABLEAU_LIBS, NB.

precond: Tous les arguments sont syntaxiquement
 et sémantiquement corrects.

postcond: Sont affichés à l'écran:

- LIBS(1) → LIBS(5)
- LIBBAR
- Les 5 premières entrées de *
 TABLEAU_VALS et la 11^e
 (entrées
 verticales)

SORTIR_LIGNES_VALS.

** Cette primitive remplit la variable LIGNE_VALEUR1 avec les valeurs appro-
 ** priées de TABLEAU_VALS (c-a-d les 5 valeurs de gauche, la valeur de code
 ** de la ligne se trouvant en 11^e position) et la sort à l'écran.

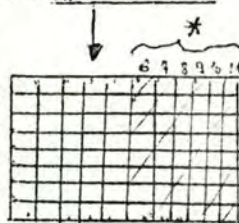
```

MOVE FLT2(Z,11) TO VAL0.
MOVE FLT2(Z,1) TO VAL1.
MOVE FLT2(Z,2) TO VAL2.
MOVE FLT2(Z,3) TO VAL3.
MOVE FLT2(Z,4) TO VAL4.
MOVE FLT2(Z,5) TO VAL5.
DISPLAY LIGNE_VALEURS1.
DISPLAY ".
  
```

SORTIR_TABLEAU2. (n°25)

** Cette primitive sort la partie de droite* d'un tableau de valeurs d'un
 ** barème à l'écran.
 ** Le titre du barème est dans LIBBAR.
 ** Les titres des colonnes se trouvent dans TABLEAU_LIBS (de longueur NB).
 ** Les valeurs se trouvent dans TABLEAU_VALS (de longueur NLI).

```
PERFORM CLEAR_SCREEN.
MOVE LIBS(6) TO BAR6.
MOVE LIBS(7) TO BAR7.
MOVE LIBS(8) TO BAR8.
MOVE LIBS(9) TO BAR9.
MOVE LIBS(10) TO BAR10.
DISPLAY LIGNE_TITRES.
DISPLAY "
DISPLAY LIGNE_TITRES.
DISPLAY LIGNE_TITRES2.
PERFORM SORTIR_LIGNE_VAL2 VARYING I FROM 6 BY 1 UNTIL I>NB.
DISPLAY "
IF NLI<03 MOVE NLI2 TO N ELSE MOVE 03 TO N.
PERFORM SORTIR_LIGNE_VAL2 VARYING Z FROM 1 BY 1 UNTIL Z>N.
DISPLAY LIGNE_TITRES.
EXIT PROGRAM.
```



Orig : TABLEAU_VALS, NLI, LIBBAR, TABLEAU_LIBS, NB

second : idem que primitive n°24

extend : Sont affichés à l'écran :

- LIBS(6) → LIBS(10)
 - LIBBAR
 - Les 5 dernières entrées de TABLEAU_VALS
- * (entrées verticales)

", LIBBAR.

SORTIR_LIGNE_VAL2.

** Cette primitive remplit la variable LIGNE_VALEUR2 avec les valeurs appro-
 ** priées de TABLEAU_VALS (c-à-d les 5 valeurs de droite des positions 6 à 10)
 ** et la sort à l'écran.

```
MOVE FLT2(Z,6) TO VAL6.
MOVE FLT2(Z,7) TO VAL7.
MOVE FLT2(Z,8) TO VAL8.
MOVE FLT2(Z,9) TO VAL9.
MOVE FLT2(Z,10) TO VAL10.
DISPLAY LIGNE_VALEURS2.
DISPLAY ".
```

 *

ALL_MESSAGE.(n°0)

** Cette primitive fait un branchement vers le message approprié a sortir
** selon la valeur le NUM_MESS.

GO TO MES01, MES02, MES03, MES04, MES05, MES06, MES07, MES08, MES09, MES10,
MES11, MES12, MES13, MES14, MES15, MES16, MES17, MES18, MES19, MES20,
MES21, MES22, MES23, MES24, MES25, MES26, MES27, MES28, MES29, MES30,
MES31, MES32, MES33, MES34, MES35, MES36, MES37, MES38, MES39, MES40,
MES41, MES42, MES43, MES44, MES45, MES46, MES47, MES48, MES49, MES50,
MES51, MES52, MES53, MES54, MES55, MES56, MES57, MES58, MES59, MES60,
MES61, MES62, MES63, MES64, MES65, MES66, MES67, MES68, MES69, MES70,
MES71, MES72, MES73, MES74, MES75, MES76, MES77, MES78, MES79, MES80,
MES81, MES82, MES83, MES84, MES85, MES86
DEPEND ON NUM_MESS.
EXIT PROGRAM.

arg: NUM_MESS
precond: 01 ≤ NUM_MESS ≤ 86
postcond:
② si num_mess = 01
alors MES01 est déclenché

** Ci-dessous se trouve donc reunis la majorite des messages du systeme.

MES01.

```
DISPLAY "*****".
DISPLAY " * MENU DU PROGRAMME * ".
DISPLAY "*****".
DISPLAY "0/ Sortir Du Programme.".
DISPLAY "".
DISPLAY "1/ Creer Un Bareme.".
DISPLAY "".
DISPLAY "2/ Creer Des Valeurs De Bareme.".
DISPLAY "".
DISPLAY "3/ Obtenir Le(s) Libelle(s) D'une Valeur De Code.".
DISPLAY "".
DISPLAY "4/ Afficher Un Bareme.".
DISPLAY "".
DISPLAY "5/ Obtenir La Composition D'un Bareme.".
DISPLAY "".
DISPLAY "6/ Obtenir Les Noms De Tous Les Baremes.".
DISPLAY "".
DISPLAY "7/ Recherche selon critere.".
DISPLAY "".
DISPLAY "8/ Obtenir les acces autorises." WITH NO ADVANCING.
DISPLAY " 9/ Nettoyage du systeme.".
DISPLAY "".
DISPLAY "".
DISPLAY " Votre reponse ? " WITH NO ADVANCING.
IF NUM_MESS NOT EQUAL 99 EXIT PROGRAM.
```

MES02.

```
DISPLAY "".
DISPLAY "Fin de creation de valeur de bareme !".
IF NUM_MESS NOT EQUAL 99 EXIT PROGRAM.
```

*

MES03.
 DISPLAY "".
 DISPLAY "Votre bareme est cree !".
 IF NUM_MESS NOT EQUAL 99 EXIT PROGRAM.

MES04.
 DISPLAY "".
 DISPLAY "Une autre creation ? " WITH NO ADVANCING.
 IF NUM_MESS NOT EQUAL 99 EXIT PROGRAM.

MES05.
 DISPLAY "".
 DISPLAY "RAPUR RETOUR ! " WITH NO ADVANCING.
 IF NUM_MESS NOT EQUAL 99 EXIT PROGRAM.

MES06.
 DISPLAY "".
 DISPLAY "Identificateur du bareme : " WITH NO ADVANCING.
 IF NUM_MESS NOT EQUAL 99 EXIT PROGRAM.

MES07.
 DISPLAY "Nombre de bar(s) composant(s) (00<=val<10) : "
 WITH NO ADVANCING.
 IF NUM_MESS NOT EQUAL 99 EXIT PROGRAM.

MES08.
 DISPLAY "Nombre de(s) libelle(s) (00<=val<40) : " WITH NO ADVANCING.
 IF NUM_MESS NOT EQUAL 99 EXIT PROGRAM.

MES09.
 DISPLAY "Date de creation JJ/MM/AA : " WITH NO ADVANCING.
 IF NUM_MESS NOT EQUAL 99 EXIT PROGRAM.

MES10.
 DISPLAY "Type des valeurs de code : " WITH NO ADVANCING.
 IF NUM_MESS NOT EQUAL 99 EXIT PROGRAM.

MES11.
 DISPLAY "Longueur maximum (00<val<07): " WITH NO ADVANCING.
 IF NUM_MESS NOT EQUAL 99 EXIT PROGRAM.

MES12.
 DISPLAY "Nbre de version (-1<val<21): " WITH NO ADVANCING.
 IF NUM_MESS NOT EQUAL 99 EXIT PROGRAM.

MES13.
 DISPLAY "Duree de retention (0000<val): " WITH NO ADVANCING.
 IF NUM_MESS NOT EQUAL 99 EXIT PROGRAM.

MES14.
 DISPLAY "Nom du bareme : " WITH NO ADVANCING.
 IF NUM_MESS NOT EQUAL 99 EXIT PROGRAM.

MES15.
 DISPLAY "".
 DISPLAY "Cet identifiant existe deja !".
 IF NUM_MESS NOT EQUAL 99 EXIT PROGRAM.

*

```

MES16.
    DISPLAY "".
    DISPLAY "Entrez le(s) bareme(s) composant(s) !".
    IF NUM_MESS NOT EQUAL 99 EXIT PROGRAM.

MES17.
    DISPLAY "".
    DISPLAY "Bareme inexistant !".
    DISPLAY "".
    IF NUM_MESS NOT EQUAL 99 EXIT PROGRAM.

MES18.
    DISPLAY "".
    DISPLAY "Violation de contrainte !".
    DISPLAY "Ce bareme est compose !".
    DISPLAY "Or un bareme composant ne peut etre que simple !".
    IF NUM_MESS NOT EQUAL 99 EXIT PROGRAM.

MES19.
    DISPLAY "".
    DISPLAY "Ce bareme est deja composant !".
    IF NUM_MESS NOT EQUAL 99 EXIT PROGRAM.

MES20.
    DISPLAY "Role du composant : " WITH NO ADVANCING.
    IF NUM_MESS NOT EQUAL 99 EXIT PROGRAM.

MES21.
    DISPLAY "".
    DISPLAY "Bareme inexistant !".
    IF NUM_MESS NOT EQUAL 99 EXIT PROGRAM.

MES22.
    DISPLAY "".
    DISPLAY "Liste des baremes composants :".
    DISPLAY "-----".
    DISPLAY "".
    IF NUM_MESS NOT EQUAL 99 EXIT PROGRAM.

MES23.
    DISPLAY "Libelle : " WITH NO ADVANCING.
    IF NUM_MESS NOT EQUAL 99 EXIT PROGRAM.

MES24.
    DISPLAY "Req.ing : " WITH NO ADVANCING.
    IF NUM_MESS NOT EQUAL 99 EXIT PROGRAM.

MES25.
    DISPLAY "Numero du bareme composant : " WITH NO ADVANCING.
    IF NUM_MESS NOT EQUAL 99 EXIT PROGRAM.

MES26.
    DISPLAY "Role du composant : " WITH NO ADVANCING.
    IF NUM_MESS NOT EQUAL 99 EXIT PROGRAM.

```

*

MES27.

```
DISPLAY "".  
DISPLAY "CE BAREME EST SIMPLE !".  
DISPLAY "".  
IF NUM_MESS NOT EQUAL 99 EXIT PROGRAM.
```

MES28.

```
DISPLAY "".  
DISPLAY "CE BAREME EST COMPOSE !".  
DISPLAY "".  
IF NUM_MESS NOT EQUAL 99 EXIT PROGRAM.
```

MES29.

```
DISPLAY "                                Valeur non admise !".  
IF NUM_MESS NOT EQUAL 99 EXIT PROGRAM.
```

MES30.

```
DISPLAY "".  
DISPLAY "Liste des baremes qui l'admettent comme composant :".  
DISPLAY "-----".  
DISPLAY "".  
IF NUM_MESS NOT EQUAL 99 EXIT PROGRAM.
```

MES31.

```
DISPLAY "".  
DISPLAY "Ce bareme n'est pas encore utilise comme composant !".  
IF NUM_MESS NOT EQUAL 99 EXIT PROGRAM.
```

MES32.

```
DISPLAY "OPTION 6: Voici le nom les baremes :".  
DISPLAY "-----".  
DISPLAY "".  
IF NUM_MESS NOT EQUAL 99 EXIT PROGRAM.
```

MES33.

```
DISPLAY "                                Votre heure est anterieure"  
DISPLAY "                                a l'heure courante!".  
IF NUM_MESS NOT EQUAL 99 EXIT PROGRAM.
```

MES34.

```
DISPLAY "".  
DISPLAY "Creation impossible car il n'existe pas de valeurs".  
DISPLAY "pour un des baremes composants.".   
DISPLAY "Veuillez les consulter obtenir plus d'informations".  
IF NUM_MESS NOT EQUAL 99 EXIT PROGRAM.
```

MES35.

```
DISPLAY "Taper G pour voir le tableau de gauche ou RETURN pour"  
WITH NO ADVANCING.  
DISPLAY " voir la suite :" WITH NO ADVANCING.  
IF NUM_MESS NOT EQUAL 99 EXIT PROGRAM.
```

MES36.

```
DISPLAY "".  
DISPLAY "Cette valeur de code existe deja !".  
DISPLAY "".  
IF NUM_MESS NOT EQUAL 99 EXIT PROGRAM.
```

*

MES37.

DISPLAY "Date de debut d'utilisation : " WITH NO ADVANCING.
IF NUM_MESS NOT EQUAL 99 EXIT PROGRAM.

MES38.

DISPLAY "Heure de debut d'utilisation : " WITH NO ADVANCING.
IF NUM_MESS NOT EQUAL 99 EXIT PROGRAM.

MES39.

DISPLAY "Date de fin d'utilisation : " WITH NO ADVANCING.
IF NUM_MESS NOT EQUAL 99 EXIT PROGRAM.

MES40.

DISPLAY "".
DISPLAY "Valeur de code inexistante !".
DISPLAY "".
IF NUM_MESS NOT EQUAL 99 EXIT PROGRAM.

MES41.

DISPLAY "".
DISPLAY "Cette valeur de code est composee !".
DISPLAY "".
IF NUM_MESS NOT EQUAL 99 EXIT PROGRAM.

MES42.

DISPLAY "".
DISPLAY "Cette valeur de code est simple !".
DISPLAY "".
IF NUM_MESS NOT EQUAL 99 EXIT PROGRAM.

MES43.

DISPLAY "Valeur de code : " WITH NO ADVANCING.
IF NUM_MESS NOT EQUAL 99 EXIT PROGRAM.

MES44.

DISPLAY "Valeur de code composante : " WITH NO ADVANCING.
IF NUM_MESS NOT EQUAL 99 EXIT PROGRAM.

MES45.

DISPLAY "Role de cette composition : " WITH NO ADVANCING.
IF NUM_MESS NOT EQUAL 99 EXIT PROGRAM.

MES46.

DISPLAY "".
DISPLAY "Liste des valeurs composantes :".
DISPLAY "-----".
DISPLAY "".
IF NUM_MESS NOT EQUAL 99 EXIT PROGRAM.

MES47.

DISPLAY "".
DISPLAY "Il n'existe encore aucun bareme !!".
IF NUM_MESS NOT EQUAL 99 EXIT PROGRAM.

*

MES18.

```
DISPLAY "".  
DISPLAY "Il n'existe pas encore de valeurs dans ce bareme !!".  
IF NUM_MESS NOT EQUAL 99 EXIT PROGRAM.
```

MES49.

```
DISPLAY "".  
DISPLAY "Ce bareme fait-il partie de la recherche ? "  
WITH NO ADVANCING.  
IF NUM_MESS NOT EQUAL 99 EXIT PROGRAM.
```

MES50.

```
DISPLAY "Vous devez entrer une valeur !!".  
IF NUM_MESS NOT EQUAL 99 EXIT PROGRAM.
```

MES51.

```
DISPLAY "Heure invalide !".  
IF NUM_MESS NOT EQUAL 99 EXIT PROGRAM.
```

MES52.

```
DISPLAY "Minute invalide !".  
IF NUM_MESS NOT EQUAL 99 EXIT PROGRAM.
```

MES53.

```
DISPLAY "Valeur non numerique !".  
IF NUM_MESS NOT EQUAL 99 EXIT PROGRAM.
```

MES54.

```
DISPLAY "Annee invalide !".  
DISPLAY "Et/ou Mois invalide !".  
IF NUM_MESS NOT EQUAL 99 EXIT PROGRAM.
```

MES55.

```
DISPLAY "Jour invalide !".  
IF NUM_MESS NOT EQUAL 99 EXIT PROGRAM.
```

MES56.

```
DISPLAY "Votre date de debut d'uti-"  
DISPLAY "lisation est anterieure a"  
DISPLAY "la date de creation !".  
IF NUM_MESS NOT EQUAL 99 EXIT PROGRAM.
```

MES57.

```
DISPLAY "Votre date de fin d'uti-"  
DISPLAY "lisation est anterieure a"  
DISPLAY "votre date de debut d'uti-"  
DISPLAY "lisation.".  
IF NUM_MESS NOT EQUAL 99 EXIT PROGRAM.
```

MES58.

```
DISPLAY "".  
DISPLAY "Taper D pour voir le tableau de droite ou RETURN pour"  
WITH NO ADVANCING.  
DISPLAY " voir la suite : " WITH NO ADVANCING.  
IF NUM_MESS NOT EQUAL 99 EXIT PROGRAM.
```

*

MES59.

```
PERFORM CLEAR_SCREEN.  
DISPLAY "                                OPTION 7: RECHERCHE SELON CRITERE."  
DISPLAY "                                -----"  
DISPLAY ""  
IF NUM_MESS NOT EQUAL 99 EXIT PROGRAM.
```

MES60.

```
DISPLAY ""  
DISPLAY "Pas de critere, donc recherche impossible !"  
IF NUM_MESS NOT EQUAL 99 EXIT PROGRAM.
```

MES61.

```
PERFORM CLEAR_SCREEN.  
DISPLAY "                                !  
DISPLAY "                                OPTION 1: Creation de bareme."  
DISPLAY "                                -----"  
DISPLAY ""  
IF NUM_MESS NOT EQUAL 99 EXIT PROGRAM.
```

MES62.

```
PERFORM CLEAR_SCREEN.  
DISPLAY "                                  
DISPLAY "                                OPTION 2: Creation de valeurs de bareme."  
DISPLAY "                                -----"  
DISPLAY ""  
IF NUM_MESS NOT EQUAL 99 EXIT PROGRAM.
```

MES63.

```
PERFORM CLEAR_SCREEN.  
DISPLAY "OPTION 3: Obtention de(s) libelle(s) d'une valeur de code."  
DISPLAY "-----"  
DISPLAY ""  
IF NUM_MESS NOT EQUAL 99 EXIT PROGRAM.
```

MES64.

```
PERFORM CLEAR_SCREEN.  
DISPLAY "                                  
DISPLAY "                                OPTION 4: Afficher un bareme."  
DISPLAY "                                -----"  
DISPLAY ""  
IF NUM_MESS NOT EQUAL 99 EXIT PROGRAM.
```

MES65.

```
PERFORM CLEAR_SCREEN.  
DISPLAY "                                  
DISPLAY "                                OPTION 5: Obtention de la composition d'un bareme."  
DISPLAY "                                -----"  
DISPLAY ""  
IF NUM_MESS NOT EQUAL 99 EXIT PROGRAM.
```

MES66.

```
DISPLAY ""  
DISPLAY "Cette valeur n'est pas encore d'actualite!"  
IF NUM_MESS NOT EQUAL 99 EXIT PROGRAM.
```

MES67.

```
DISPLAY ""  
DISPLAY "Cette valeur n'est plus d'actualite!"  
IF NUM_MESS NOT EQUAL 99 EXIT PROGRAM.
```

*

MES68.

```
DISPLAY "".  
DISPLAY "Recherche sur critere interlité pour les baremes simples!".  
IF NUM_MESS NOT EQUAL 99 EXIT PROGRAM.
```

MES69.

```
DISPLAY "".  
DISPLAY "Aucune ligne de ce bareme ne correspond a votre critere!".  
IF NUM_MESS NOT EQUAL 99 EXIT PROGRAM.
```

MES70.

```
DISPLAY "".  
DISPLAY "".  
DISPLAY "Desole, vous n'avez pas ete reconnu par le systeme !".  
IF NUM_MESS NOT EQUAL 99 EXIT PROGRAM.
```

MES71.

```
PERFORM CLEAR_SCREEN.MOVE ID_UTIL TO NUM_UTIL.  
DISPLAY "" DISPLAY "" DISPLAY "" DISPLAY "" DISPLAY "".  
DISPLAY "" DISPLAY "".  
DISPLAY "                                BONJOUR, ", NUM_UTIL.  
DISPLAY "".  
DISPLAY "                                bienvenue sur GESEBAR !".  
DISPLAY "".  
DISPLAY "                                Bon travail!".  
DISPLAY "".  
DISPLAY "                                Taper RETURN : "  
WITH NO ADVANCING.  
IF NUM_MESS NOT EQUAL 99 EXIT PROGRAM.
```

MES72.

```
PERFORM CLEAR_SCREEN.  
DISPLAY "" DISPLAY "" DISPLAY "".  
DISPLAY "*****".  
DISPLAY "                                *".  
DISPLAY "                                * SYSTEME DE GESTION DE BAREMES *".  
DISPLAY "                                *".  
DISPLAY "*****".  
DISPLAY "" DISPLAY "".  
DISPLAY "Memoire pour le Credit Communal de Bruxelles."  
DISPLAY "" DISPLAY "".  
DISPLAY "Systeme Cree Par E.EVRARD, le 10/04/1986".  
DISPLAY "" DISPLAY "".  
DISPLAY "Bon l'utilisateur : " WITH NO ADVANCING.  
IF NUM_MESS NOT EQUAL 99 EXIT PROGRAM.
```

MES73.

```
DISPLAY ""  
DISPLAY "Mot de passe : " WITH NO ADVANCING.  
IF NUM_MESS NOT EQUAL 99 EXIT PROGRAM.
```

MES74.

```
DISPLAY "".  
DISPLAY "Vous n'avez pas acces a ce bareme !".  
IF NUM_MESS NOT EQUAL 99 EXIT PROGRAM.
```

*

MES75.

```
PERFORM CLEAR_SCREEN.  
DISPLAY "                                OPTION 3: ACCES AUTORISES."  
DISPLAY "                                -----".  
DISPLAY "".  
IF NUM_MESS NOT EQUAL 99 EXIT PROGRAM.
```

MES76.

```
DISPLAY "Le type de la valeur doit etre ",TP_VAL.  
IF NUM_MESS NOT EQUAL 99 EXIT PROGRAM.
```

MES77.

```
DISPLAY "La longueur maximum de la valeur est ",LG_VAL.  
IF NUM_MESS NOT EQUAL 99 EXIT PROGRAM.
```

MES78.

```
DISPLAY "".  
DISPLAY "Vous n'avez encore aucun acces !".  
IF NUM_MESS NOT EQUAL 99 EXIT PROGRAM.
```

MES79.

```
DISPLAY ".DISPLAY".DISPLAY".DISPLAY".DISPLAY".  
DISPLAY ".DISPLAY".DISPLAY".DISPLAY".DISPLAY".  
DISPLAY "Voulez - vous voir les entetes".  
DISPLAY "des barenes composants <0 ou H> ? "WITH NO ADVANCING.  
IF NUM_MESS NOT EQUAL 99 EXIT PROGRAM.
```

MES80.

```
DISPLAY "".  
DISPLAY "Veuillez consulter ce(s) barene(s) !".  
IF NUM_MESS NOT EQUAL 99 EXIT PROGRAM.
```

MES81.

```
PERFORM CLEAR_SCREEN.  
DISPLAY "                                OPTION 9: Nettoyage du systeme .".  
DISPLAY "                                -----".  
DISPLAY "".  
IF NUM_MESS NOT EQUAL 99 EXIT PROGRAM.
```

MES82.

```
DISPLAY "".  
DISPLAY "Liste des barenes supprimees:".  
DISPLAY "-----".  
IF NUM_MESS NOT EQUAL 99 EXIT PROGRAM.
```

*

MES83.

DISPLAY "Aucun bareme n'est a supprimer!".
IF NUM_MESS NOT EQUAL 99 EXIT PROGRAM.

MES84.

DISPLAY "Ce libelle existe deja. Donnez en un autre! Merci!".
IF NUM_MESS NOT EQUAL 99 EXIT PROGRAM.

MES85.

DISPLAY "Val.poss : AN pour anglais,".
DISPLAY " FR pour francais,".
DISPLAY " IR pour neerlandais,".
DISPLAY " AL pour allemand,".
IF NUM_MESS NOT EQUAL 99 EXIT PROGRAM.

MES86.

DISPLAY "Tapez S pour stopper ! " WITH NO ADVANCING.
IF NUM_MESS NOT EQUAL 99 EXIT PROGRAM.

Programme SECURITE.COB

IDENTIFICATION DIVISION.

PROGRAM-ID. MODULE_SECURITE.

AUTHOR. E EVRARD.
DATE-WRITTEN. 16/04/1986.

**
** MODULE SECURITE **
**

** Ce module a pour but de realiser les primitives qui touchent la securite
** du systeme. Il renferme le secret de la composition du schema de la
** securite. Actuellement ce schema est tres simplifie. Il aurait fallu
** plus de temps pour developper le schema en entier et un ensemble complet
** de primitives travaillant dessus. Le but est donc ici de montrer un
** exemple de ce qu'on peut faire. ce module est bien entendu extensible
** comme on le desire.

ENVIRONMENT DIVISION.

CONFIGURATION SECTION.
*-----

SOURCE-COMPUTER. vax_11.
OBJECT-COMPUTER. vax_11.

DATA DIVISION.

SUB-SCHEMA SECTION.
*-----

DB DEFAULT_SUBSCHEMA WITHIN GESTBAR FOR "GESTBAR.ROO".
LD KEEP_LIST.

WORKING-STORAGE SECTION.
*-----

01 FINI PIC 9(1).
01 I PIC 9(2).

LINKAGE SECTION.
*-----

01 NUM_PROC PIC 9(2).
01 ID_UTIL PIC X(20).
01 ID_BAR PIC X(6).
01 EXISTE PIC 9(1).
01 TAB.
 02 NR PIC X(6) OCCURS 99 TIMES.
01 N PIC 9(2).
*


```
*****
PROCEDURE DIVISION USING NUM_PROC, ID_UTIL, ID_BAR, TAB, N, EXISTE.
*****
```

```
*****
DECLARATIVES.
```

```
** Voici ici le morceau de programme qui est appele automatiquement si
** une erreur de base de donnees se produit.
```

```
100-DATABASE-EXCEPTION SECTION.
    USE FOR DB-EXCEPTION ON OTHER.
100-PROCEDURE.
    DISPLAY "DATABASE EXCEPTION CONDITION".
    PERFORM 150-DISPLAY-MESSAGE.
```

```
150-DISPLAY-MESSAGE.
    CALL "DBM$SIGNAL".
    ROLLBACK.
    STOP RUN.
```

```
END DECLARATIVES.
```

```
*****
NIVEAU1 SECTION.
*-----
```

```
*****
APPEL_PROCEDURE.
```

```
** Ici se trouve le programme general qui effectue un branchement vers la
** primitive appropriee selon la valeur de NUM_PROC.
```

```
IF NUM_PROC = 01 PERFORM EXIST_UTILISATEUR.
IF NUM_PROC = 02 PERFORM EXIST_ACCES.
IF NUM_PROC = 03 PERFORM CREER_ACCES.
IF NUM_PROC = 04 PERFORM GIVE_ACCES.
```

```
*****
*
```


NIVEAU2 SECTION.

EXIST_UTILISATEUR.

** Cette primitive controle s'il existe un article UTILISATEUR dont la valeur
** d'item est ID_UTIL. Si oui, EXISTE doit valoir 1 sinon 0.

```
READY CONCURRENT RETRIEVAL.  
MOVE ID_UTIL TO NR_UTIL.  
MOVE 1 TO EXISTE.  
FETCH FIRST UTILISATEUR USING NR_UTIL  
    AT END MOVE 0 TO EXISTE.  
ROLLBACK.  
EXIT PROGRAM.
```

EXIST_ACCES.

** Cette primitive verifie si l'article UTILISATEUR de cle ID_UTIL possede
** dans son chemin LIEN_ACC_BAR un article BAREME dont le numero est ID_BAR.
** Si oui, EXISTE doit valoir 1 sinon 0.

```
READY CONCURRENT RETRIEVAL.  
MOVE ID_UTIL TO NR_UTIL.  
FETCH FIRST UTILISATEUR USING NR_UTIL.  
MOVE 0 TO EXISTE.MOVE 0 TO FINI.  
PERFORM VERIF_ACCES UNTIL FINI=1.  
ROLLBACK.  
EXIT PROGRAM.
```

CREER_ACCES.

** Cette primitive doit inserer dans le chemin LIEN_ACC_BAR dont l'owner est
** l'article UTILISATEUR de numero ID_UTIL, l'article BAREME dont le numero
** est ID_BAR.

```
READY PROTECTED UPDATE.  
MOVE ID_UTIL TO NR_UTIL.  
MOVE ID_BAR TO NR_BAR.  
FIND FIRST UTILISATEUR USING NR_UTIL.  
FIND FIRST BAREME USING NR_BAR.  
CONNECT TO LIEN_ACCES_BAR.  
COMMIT  
EXIT PROGRAM.
```

*

GIVE_ACCES.

** Cette primitive doit accéder dans le chemin LIEN_ACC_BAR dont l'owner
** est l'article UTILISATEUR de numero ID_UTIL, aux articles BAREME membres
** de ce chemin afin d'en memoriser les valeurs de l'item NR_BAR.

READY CONCURRENT RETRIEVAL.
MOVE ID_UTIL TO NR_UTIL.
FIND FIRST UTILISATEUR USING NR_UTIL.
MOVE 0 TO FINI.
PERFORM OBT_ACCES VARYING I FROM 1 BY 1 UNTIL FINI=1.
ROLLBACK.
EXIT PROGRAM.

NIVEAU3 SECTION.

OBT_ACCES.

** Corp de la boucle de la primitive GIVE_ACCES

FETCH NEXT BAREME WITHIN LIEN_ACCES_BAR
AT END MOVE 1 TO FINI
COMPUTE N = I - 1.
IF FINI=0
MOVE NR_BAR TO NR(I).

VERIF_ACCES.

** Corp de la boucle de la primitive EXIST_ACCES

FETCH NEXT BAREME WITHIN LIEN_ACCES_BAR
AT END MOVE 1 TO FINI.
IF FINI=0 AND NR_BAR=ID_BAR
MOVE 1 TO EXISTE
MOVE 1 TO FINI.

4/ Les outils.

Programme INTERS-DATE.COB

IDENTIFICATION DIVISION.

PROGRAM-ID. INTERS-DATE.

AUTHOR. EFVDAPP.
DATE-WRITTEN. AVRIL 86.

**
** INTERSECTION D'UNE SERIE DE DATES **
**

** Ce programme doit fournir dans DT1 la date superieure des dates du champ
** DAT-DBT du tableau TAB-DATES, dans DT2 la date inferieure des dates du
** champ DAT-FIN du tableau TAB-DATES, et dans H l'heure superieure des
** heures du champ HEU-DBT du tableau TAB-DATES.

ENVIRONNEMENT DIVISION.

CONFIGURATION SECTION.

SOURCE-COMPUTER. vax-11.
OBJECT-COMPUTER. vax-11.

DATA DIVISION.

WORKING-STORAGE SECTION.

01 I PIC 9(2).
01 ETAT PIC X(3).

LINKAGE SECTION.

01 TAB-DATES.
02 ELEMENT OCCURS 1 TO 10 TIMES DEPENDING ON I.
03 DAT-DBT PIC X(10).
03 HEU-DBT PIC X(5).
03 DAT-FIN PIC X(10).

01 H PIC 9(2).
01 DT1 PIC X(10).
01 DT2 PIC X(10).
01 J PIC X(5).
*


```

*****
PROCEDURE DIVISION USING TAB_DATES, I, DT1, H, DT2.
*****

```

NIVEAU1 SECTION.

PROG_TEST.

** Voici le programme general qui nous donnera les 3 arguments de sorties
 ** DT1,DT2 et H tels que decrits ci-dessus.

```

  MOVE DAT_DBT(1) TO DT1.
  PERFORM OBTENIR_DT_SUP VARYING I FROM 2 BY 1 UNTIL I>N.
  MOVE HEU_DBT(1) TO H.
  PERFORM OBTENIR_HR_SUP VARYING I FROM 2 BY 1 UNTIL I>N.
  MOVE DAT_FIN(1) TO DT2.
  PERFORM OBTENIR_DT_INF VARYING I FROM 2 BY 1 UNTIL I>I.
  EXIT PROGRAM.

```

NIVEAU2 SECTION.

OBTENIR_DT_SUP.

** Si DAT_DBT (I) est superieur a DT1, il doit remplacer la valeur de ce DT1
 ** puisqu'on veut la date maximum.

```

  CALL "COMPARE_DATE" USING DAT_DBT(I),DT1,ETAT.
  IF ETAT="SUP"
    MOVE DAT_DBT(I) TO DT1.

```

OBTENIR_DT_INF.

** Si DAT_FIN (I) est inferieur a DT2, il doit remplacer la valeur de ce DT2
 ** puisqu'on veut la date minimum.

```

  CALL "COMPARE_DATE" USING DAT_FIN(I),DT2,ETAT.
  IF ETAT="INF"
    MOVE DAT_FIN(I) TO DT2.

```

OBTENIR_HR_SUP.

** Si HEU_DBT (I) est superieur a H, il doit remplacer la valeur de ce H
 ** puisqu'on veut l'heure maximum.

```

  CALL "COMPARE_HEURE" USING HEU_DBT(I),H,ETAT.
  IF ETAT="SUP"
    MOVE HEU_DBT(I) TO H.

```

Programme CONTROL-CODE .COB

IDENTIFICATION DIVISION.

PROGRAM-ID. CONTROL_CODE.

AUTHOR. E. EVRARD.

DATE-WRITTEN. 10/05/1986.

**
** CONTROLLE DE LONGUEUR ET DE TYPE D'UNE VALEUR DE CODE **
**

** Ce programme a pour but de controler si la valeur de code VAL_CODE
** est du type TP_VAL et de longueur maximum LG_VAL. Si le type est
** incorrect, STAT1 doit valoir 1 sinon 0. Si la longueur est incorrecte,
** STAT2 doit valoir 1 sinon 0.

ENVIRONNEMENT DIVISION.

CONFIGURATION SECTION.

SOURCE-COMPUTER. vax-11.
OBJECT-COMPUTER. vax-11.

DATA DIVISION.

WORKING-STORAGE SECTION.

01 I PIC X.
01 VCODE.
 02 C1 PIC X.
 02 C2 PIC X.
 02 C3 PIC X.
 02 C4 PIC X.
 02 C5 PIC X.
 02 C6 PIC X.
01 TAB_CAP; REDEFINES VCODE.
 02 FLT; DECRES 6 TIMES.
 03 CAR PIC X.
01 E PIC 9(2).
01 HALT PIC 9(1).
01 L PIC 9(2).
*

LINKAGE SECTION.

```
01 VAL_CODE PIC X(6).
01 LG_VAL PIC 9(2).
01 TP_VAL PIC X(2).
01 ETAT1 PIC 9(1).
01 ETAT2 PIC 9(1).
```

```
*****
PROCEDURE DIVISION USING VAL_CODE, LG_VAL, TP_VAL, ETAT1, ETAT2.
*****
```

NIVEAU1 SECTION.

```
*****
*****
```

PROGRAMME.

```
** Voici le programme general qui calcule la longueur effective de VAL_CODE,
** effectue les controles prevus le type et de longueur et retourne des valeurs
** appropriees dans ETAT1 et ETAT2.
```

```
    MOVE VAL_CODE TO VCODE.
    PERFORM INIT.
    MOVE 0 TO HALT.
    PERFORM CALCUL_LONG VARYING I FROM 6 BY -1 UNTIL HALT=1.
    PERFORM CONTROL_TYPE.
    PERFORM CONTROL_LONGUEUR.
    EXIT PROGRAM.
```

```
*****
*****
```

NIVEAU2 SECTION.

```
*****
*****
```

INIT.

```
** Initialisation des variables ETAT1 et ETAT2.
```

```
    MOVE 0 TO ETAT1.
    MOVE 0 TO ETAT2.
```

```
*****
*
```

CALCUL_LONG.

** Corps de la boucle du programme general qui a pour but de calculer la
** longueur le VAL_CODE.

```
IF I = 0  MOVE I TO L
          MOVE 1 TO HALT
ELSE IF CAR(I) DOT = "" MOVE 1 TO HALT
          MOVE I TO L
END-IF.
```

CONTROL_LONGUEUR.

** Cette primitive met 1 dans ETAT2 seulement si la longueur de VAL_CODE
** est plus grande que LG_VAL.

```
IF L > LG_VAL MOVE 1 TO ETAT2.
```

CONTROL_TYPE.

** Cette primitive doit contrôler le type de VAL_CODE.

```
IF TP_VAL = "NU"
  PERFORM BOUCLE_NUM VARYING I FROM 1 BY 1 UNTIL I > L.
IF TP_VAL = "AB"
  PERFORM BOUCLE_TYPE VARYING I FROM 1 BY 1 UNTIL I > L.
```

NIVEAU3 SECTION.

BOUCLE_NUM.

** Corps de boucle de la primitive CONTROL_TYPE.

```
IF CAR(I) NOT NUMERIC MOVE 1 TO ETAT1.
```

BOUCLE_TYPE.

** Corps de boucle de la primitive CONTROL_TYPE.

```
IF CAR(I) < "A" OR CAR(I) > "E"  MOVE 1 TO ETAT1.
IF CAR(I) = " " OR CAR(I) = "." MOVE 0 TO ETAT1.
```

Programme CALCUL-DATE.COB

IDENTIFICATION DIVISION.

PROGRAM-ID. CALCUL-DATE.

AUTHOR. ENVIRAPD.
DATE-WRITTEN. JULY 86.

**
**
**
**

CALCUL D'UNE DATE
**
**
**

** Ce programme doit retourner dans DATE2 une date qui correspond a celle qui
** se trouve dans DATE a laquelle on a ajoute le nombre de jours qui se trouve
** dans DUREE.

ENVIRONMENT DIVISION.

CONFIGURATION SECTION.

SOURCE-COMPUTER. vax-11.
OBJECT-COMPUTER. vax-11.

DATA DIVISION.

WORKING-STORAGE SECTION.

01 DETER.
 02 JR PIC 9(2).
 02 FILLER; PIC X; VALUE "/".
 02 MS; PIC 9(2).
 02 FILLER; PIC X; VALUE "/".
 02 AN; PIC 9(1).

01 JOUR PIC 9(4); VALUE 0.
01 JOURX; REDEFINES JOUR.
 02 J1 PIC 9(2).
 02 J2 PIC 9(2).

01 RESUL PIC 9(3).
01 RESPT PIC 9(2).
01 GROPE PIC 9(1).

*

LINKAGE SECTION.

01 DATE1 PIC X(10).

01 DUREE PIC 9(4).

01 DATE2 PIC X(10).

PROCEDURE DIVISION USING DATE1, DUREE, DATE2.

N1 SECTION.

PROGRAM.

** Voici le programme general. Le principe est de reprendre DATE1,
 ** l'ajouter DUREE a son nombre de jours et de faire diminuer successi-
 ** venant le resultat de cette addition en faisant avancer le mois(modulo 12)
 ** et l'annee chaque fois que necessaire. C'est la fin de ce calcul des
 ** que l'on arrive a un mois dont le nombre de jours restant est inferieur
 ** ou egal au nombre maximum de jours pour ce mois.

MOVE DATE1 TO DT-TR.

MOVE 0 TO STOPPE.

MOVE JR TO JOUR.

COMPUTE JOUR = JOUR + DUREE.

DIVIDE AN BY 4 GIVING RESUL REMAINDER RESTE.

IF (MS=01 OR MS=03 OR MS=05 OR MS=07 OR MS=08 OR MS=10 OR MS=12)
 AND (JOUR<32) MOVE 1 TO STOPPE MOVE J2 TO JR.

IF (MS=04 OR MS=06 OR MS=09 OR MS=11) AND (JOUR<31)
 MOVE 1 TO STOPPE MOVE J2 TO JR.

IF MS=02 AND RESTE=00 AND JOUR<30 MOVE 1 TO STOPPE MOVE J2 TO JR.

IF MS=02 AND RESTE>00 AND JOUR<29 MOVE 1 TO STOPPE MOVE J2 TO JR.

IF STOPPE=0 PERFORM CALCUL UNTIL STOPPE = 1.

MOVE DT-TR TO DATE2.

EXIT PROGRAM.

 *

N2 SECTION.

CALCUL.

```
IF MS=01 OR MS=03 OR MS=05 OR MS=07 OR MS=08 OR MS=10 OR MS=12
  COMPUTE JOUR = JOUR - 31.
IF MS=01 OR MS=06 OR MS=09 OR MS=11
  COMPUTE JOUR = JOUR - 30.
DIVIDE AN BY 4 GIVING RESUL REMAINDER RESTE.
IF MS=02 AND RESTE=00 COMPUTE JOUR = JOUR - 29.
IF MS=02 AND RESTE>00 COMPUTE JOUR = JOUR - 28.
COMPUTE MS = MS + 1.
IF MS=13 MOVE 01 TO MS COMPUTE AN = AN + 1.
IF (MS=01 OR MS=03 OR MS=05 OR MS=07 OR MS=08 OR MS=10 OR MS=12)
  AND (JOUR<32) MOVE 1 TO STOPPE MOVE J2 TO JR.
IF (MS=04 OR MS=06 OR MS=09 OR MS=11) AND (JOUR<31)
  MOVE 1 TO STOPPE MOVE J2 TO JR.
IF MS=02 AND RESTE=00 AND JOUR<30 MOVE 1 TO STOPPE MOVE J2 TO JR.
IF MS=02 AND RESTE>00 AND JOUR<29 MOVE 1 TO STOPPE MOVE J2 TO JR.
```


Programme TIMER.COB

**

** Ce programme a pour but l'accéder a l'heure et la date courante
** du systeme. la date sera renvoyee dans l'argument TIME_FIELD et
** l'heure dans HOUR_FIELD.

ENVIRONMENT DIVISION.

CONFIGURATION SECTION.

SOURCE-COMPUTER. vax_11.

OBJECT-COMPUTER. vax_11.

DATA DIVISION.

WORKING-STORAGE SECTION.

01 H_DATE.

02 H_DAY; PIC 99.

02 FILLER; PIC X; VALUE "/".

02 H_MON; PIC 99.

02 FILLER; PIC X; VALUE "/".

02 FILLER; PIC 99; VALUE 19.

02 H_YEA; PIC 99.

*

```

01 H-HEURE.
    02 H_HOU; PIC 99.
    03 H_MIN; PIC 99; VALUE "1".
    03 H_SEC; PIC 99.

```

```

01 TAMPON-DATE.
    02 T_YEAR; PIC 99.
    02 T_MON; PIC 99.
    02 T_DAY; PIC 99.

```

```

01 TAMPON-HEURE.
    02 T_HOU; PIC 99.
    02 T_MIN; PIC 99.
    02 T_SEC; PIC 99.
    02 T_CN; PIC 99.

```

LINKAGE SECTION.

```

*-----

```

```

01 TIME-FIELD PIC X(10).
01 HOUR-FIELD PIC X(5).

```

```

*****
PROCEDURE DIVISION USING TIME-FIELD, HOUR-FIELD.
*****

```

SET-TIME.

```

*****

```

```

ACCEPT TAMPON-DATE FROM DATE.
ACCEPT TAMPON-HEURE FROM TIME.
MOVE CORRESPONDING TAMPON-DATE TO H-DATE.
MOVE CORRESPONDING TAMPON-HEURE TO H-HEURE.
MOVE H-DATE TO TIME-FIELD.
MOVE H-HEURE TO HOUR-FIELD.
EXIT PROGRAM.

```

```

*****
*****

```


Programme COMP-DATE.COB

IDENTIFICATION DIVISION.

PROGRAM-ID. COMPARE-DATE.

AUTHOR. LEVYARD.
DATE-WRITTEN. AVRIL 86.

**
** COMPARAISON DE DEUX DATES **
**

** Ce programme doit comparer les 2 dates DATE1 et DATE 2.
** STAT vaudra "INF" si DATE1 est inferieure a DATE2.
** " " SUP " superieure " .
** " " EGA " egale " .

ENVIRONMENT DIVISION.

CONFIGURATION SECTION.

SOURCE-COMPUTER. vax.11.
OBJECT-COMPUTER. vax.11.

DATA DIVISION.

WORKING-STORAGE SECTION.

01 H-DATE-UN.
02 H-DAY-UN; PIC 99.
02 FILLER; PIC X; VALUE "/".
02 H-MON-UN; PIC 99.
02 FILLER; PIC X; VALUE "/".
02 H-YEA-UN; PIC 9999.

01 H-DATE-DE.
02 H-DAY-DE; PIC 99.
02 FILLER; PIC X; VALUE "/".
02 H-MON-DE; PIC 99.
02 FILLER; PIC X; VALUE "/".
02 H-YEA-DE; PIC 9999.

MESSAGE SECTION.

01 DATE_UN PIC X(10).

01 DATE_DE PIC X(10).

01 ETAT PIC X(3).

PROCEDURE D'INSTRUCTION: DATE_UN, DATE_DE, ETAT.

COMPARAISON.

** Le principe est de comparer l'abord les années puis les mois puis
** les jours si nécessaire.

MOVE DATE_UN TO H-DATE_UN.

MOVE DATE_DE TO H-DATE_DE.

IF H-YEAR_UN > H-YEAR_DE
MOVE "SUP" TO ETAT
EXIT PROGRAM.

IF H-YEAR_UN = H-YEAR_DE
AND H-MON_UN > H-MON_DE
MOVE "SUP" TO ETAT
EXIT PROGRAM.

IF H-YEAR_UN = H-YEAR_DE
AND H-MON_UN = H-MON_DE
AND H-DAY_UN > H-DAY_DE
MOVE "SUP" TO ETAT
EXIT PROGRAM.

IF H-YEAR_UN = H-YEAR_DE
AND H-MON_UN = H-MON_DE
AND H-DAY_UN = H-DAY_DE
MOVE "EGA" TO ETAT
EXIT PROGRAM.

MOVE "INF" TO ETAT.
EXIT PROGRAM.

Programme COMP-HEURE.COB

IDENTIFICATION DIVISION.

PROGRAM-ID. COMPARE-HEURES.

AUTHOR. SEVERARD.

DATE-WRITTEN. AVRIL 66.

**

**

**

COMPARAISON DE DEUX HEURES

**

**

**

** Ce programme doit comparer 2 heures HEURE1 et HEURE2.

** ETAT doit valoir "INF" si HEURE1 est inferieure a HEURE2.

**	"	"	SUP	"	superieure	"	:
**	"	"	EGA	"	egale	"	:

ENVIRONNEMENT DIVISION.

CONFIGURATION SECTION.

SOURCE-COMPUTER. vax_11.

OBJECT-COMPUTER. vax_11.

DATA DIVISION.

WORKING-STORAGE SECTION.

01 H1_FIELD.

02 CHP_H11 PIC X(2).

02 CHP_FL PIC X(1).

02 CHP_MH1 PIC X(2).

01 H2_FIELD.

02 CHP_H22 PIC X(2).

02 CHP_FL PIC X(1).

02 CHP_MH2 PIC X(2).

LINKAGE SECTION.

01 HEURE1 PIC X(5).

01 HEURE2 PIC X(5).

01 ETAT PIC X(3).

*

```
*****
PROCEDURE DIVISION USING HEURE1, HEURE2, ETAT.
*****
```

```
COMPARE-TIME.
*****
```

```
** Le principe est de comparer d'abord les heures et puis les minutes
** si c'est nécessaire.
```

```
MOVE HEURE1 TO H1-FIELD.
MOVE HEURE2 TO H2-FIELD.
```

```
IF CMP_HR1 > CMP_HR2
MOVE "SJP" TO ETAT
EXIT PROGRAM.
```

```
IF CMP_HR1 = CMP_HR2
AND CMP_MIN1 > CMP_MIN2
MOVE "SJP" TO ETAT
EXIT PROGRAM.
```

```
IF CMP_HR1 = CMP_HR2
AND CMP_MIN1 = CMP_MIN2
MOVE "EJA" TO ETAT
EXIT PROGRAM.
```

```
MOVE "INF" TO ETAT.
EXIT PROGRAM.
```

```
*****
*****
```


5/ La base de données.

SCHEMA NAME IS GSDTAR

AREA NAME IS PRINCIPAL

RECORD NAME IS DATE11

WITHIN PRINCIPAL

ITEM NAME IS DATE_CREATION

TYPE IS CHARACTER 10

ITEM NAME IS NR_BAR

TYPE IS CHARACTER 6

ITEM NAME IS TYPE_VAL

TYPE IS CHARACTER 2

DEFAULT VALUE IS "A1"

ITEM NAME IS LONG_VAL

TYPE IS CHARACTER 2

ITEM NAME IS NBR_VERS_GARD

TYPE IS CHARACTER 2

ITEM DUP_PETENTID1

TYPE IS CHARACTER 4

RECORD NAME IS LIB_BAR

WITHIN PRINCIPAL

ITEM NAME IS VAL_LIB_BAR

TYPE IS CHARACTER 50

ITEM NAME IS CODE_RD_BAR

TYPE IS CHARACTER 2

RECORD NAME IS LIEN_BAR

WITHIN PRINCIPAL

ITEM NAME IS NR_BAR_LB

TYPE IS CHARACTER 6

ITEM NAME IS ROLE_LB

TYPE IS CHARACTER 30

RECORD NAME IS REL_BAR_CODE

WITHIN PRINCIPAL

ITEM NAME IS DATE_CREATION

TYPE IS CHARACTER 10

ITEM DATE_DEB_UTIL

TYPE IS CHARACTER 10

ITEM HEURE_DEB_UTIL

TYPE IS CHARACTER 5

ITEM NAME IS DATE_FIN_UTIL

TYPE IS CHARACTER 10

ITEM NAME IS NRBAR_VALCODE

TYPE IS CHARACTER 12

ITEM NAME IS NRBAR_REL

TYPE IS CHARACTER 6

RECORD NAME IS LIB_VAL_CODE

WITHIN PRINCIPAL

ITEM NAME IS VAL_LIB_LVT

TYPE IS CHARACTER 50

ITEM NAME IS CODE_RD_LVT

TYPE IS CHARACTER 2

RECORD NAME IS LIEN_REL

WITHIN PRINCIPAL
ITEM NAME IS RELC_LBR
TYPE IS CHARACTER 80
ITEM NAME IS RELC_COMP_LBR
TYPE IS CHARACTER 12

RECORD NAME IS UTILISATEUR
WITHIN PRINCIPAL
ITEM NAME IS NO_UTIL
TYPE IS CHARACTER 20

SET NAME IS LIEN_ACCES_BAR
** Attention : CECI EST LE LIEN "AUTEUR" DE NOTRE SCHEMA
OWNER IS UTILISATEUR
MEMBER IS BARNE
INSERTION IS MANUAL RETENTION IS FIXED

SET NAME IS ALL_UTIL
OWNER IS SYSTEM
MEMBER IS UTILISATEUR
INSERTION IS AUTOMATIC RETENTION IS FIXED

SET NAME IS ALL_BAR
OWNER IS SYSTEM
MEMBER IS BARNE
INSERTION IS AUTOMATIC RETENTION IS FIXED

SET NAME IS LIEN_BAR_LIB
OWNER IS BARNE
MEMBER IS LIB_BAR
INSERTION IS AUTOMATIC RETENTION IS FIXED
ORDER IS NEXT

SET NAME IS EST_COMP_DE
OWNER IS BARNE
MEMBER IS LIEN_BAR
INSERTION IS AUTOMATIC RETENTION IS FIXED
ORDER IS NEXT

SET NAME ALL_PEL
OWNER IS SYSTEM
MEMBER IS REL_BAR_CODE
INSERTION IS AUTOMATIC RETENTION IS FIXED

SET NAME IS LIEN_RSC
OWNER IS REL_BAR_CODE
MEMBER IS LIEN_RSC
INSERTION IS AUTOMATIC RETENTION IS FIXED
ORDER IS NEXT

SET NAME IS LIEN_RSC_LIB
OWNER IS REL_BAR_CODE
MEMBER IS LIB_VAL_CODE

INSERTION IS AUTOMATIC RETENTION IS FIXED
ORDER IS NEXT

SET NAME ALL_LIP_BAR
OWNER IS SYSTEM
MEMBER IS LIP_BAR
INSERTION IS AUTOMATIC RETENTION IS FIXED

SET NAME ALL_LIEN_BAR
OWNER IS SYSTEM
MEMBER IS LIEN_BAR
INSERTION IS AUTOMATIC RETENTION IS FIXED

SET NAME ALL_LIEN_RED
OWNER IS SYSTEM
MEMBER IS LIEN_RED
INSERTION IS AUTOMATIC RETENTION IS FIXED

* CDD path to schema is "CDD\$TOP.IIF.WM.EVVRARD.GCS\$BAR"

*-----

STORAGE SCHEMA NAME IS DEFAULT_STORAGE_SCHEMA FOR GCS\$BAR SCHEMA

RECORD NAME IS BAR\$B

PLACEMENT IS CLUSTERED VIA ALL\$BAR

ITEM DATE_CREATED TYPE IS CHARACTER 10

ALLOCATION IS STATIC

ITEM NR\$BAR TYPE IS CHARACTER 6

ALLOCATION IS STATIC

ITEM TYPE_VAL TYPE IS CHARACTER 2

ALLOCATION IS STATIC

ITEM LONG_VAL TYPE IS CHARACTER 2

ALLOCATION IS STATIC

ITEM NBR_VERS\$BAR TYPE IS CHARACTER 2

ALLOCATION IS STATIC

ITEM CDR_IDENTIFY TYPE IS CHARACTER 4

ALLOCATION IS STATIC

RECORD NAME IS LIB\$BAR

PLACEMENT IS CLUSTERED VIA LIEN\$BAR_LIB

ITEM VAL_LIB\$BAR TYPE IS CHARACTER 50

ALLOCATION IS DYNAMIC

ITEM CODE_REL\$BAR TYPE IS CHARACTER 2

ALLOCATION IS STATIC

RECORD NAME IS LIEN\$BAR

PLACEMENT IS CLUSTERED VIA SET EST_COMP_DE

ITEM NR\$BAR_LIB TYPE IS CHARACTER 6

ALLOCATION IS STATIC

ITEM ROLE_LIB TYPE IS CHARACTER 30

ALLOCATION IS DYNAMIC

RECORD NAME IS REL\$BAR_CODE

PLACEMENT IS CLUSTERED VIA ALL\$REL

ITEM DATE_CREATION TYPE IS CHARACTER 10

ALLOCATION IS STATIC

ITEM DATE_DES_UTIL TYPE IS CHARACTER 10

ALLOCATION IS STATIC

ITEM HEURE_DES_UTIL TYPE IS CHARACTER 5

ALLOCATION IS STATIC

ITEM DATE_FIN_UTIL TYPE IS CHARACTER 10

ALLOCATION IS STATIC

ITEM NBR\$VALCODE TYPE IS CHARACTER 12

ALLOCATION IS STATIC

ITEM NBR\$REL TYPE IS CHARACTER 6

ALLOCATION IS STATIC

RECORD NAME IS LIB_VAL_CODE

PLACEMENT IS CLUSTERED VIA LIEN\$REL_LIB

ITEM VAL_LIB_LVT TYPE IS CHARACTER 50

ALLOCATION IS DYNAMIC

ITEM CODE_REL_LVT TYPE IS CHARACTER 2

ALLOCATION IS STATIC

RECORD NAME IS LIEN_LIB

PLACEMENT IS CLUSTERED VIA LIEN\$REL

ITEM ROLE_LIB TYPE IS CHARACTER 30

ALLOCATION IS DYNAMIC

ITEM REL_COMP_LBR TYPE IS CHARACTER 12
ALLOCATION IS STATIC

RECORD NAME IS UTILISATEUR
ITEM NAME IS JR_UTIL
TYPE IS CHARACTER 20

SET NAME IS LIEN_ACCES_BAR
MODE IS CHAIN

SET NAME IS ALL_BAR
MODE IS CALC
MEMBER IS DAREME
KEY IS JR_BAR

SET NAME IS LIEN_BAR_LIB
MODE IS CHAIN

SET NAME IS EST_CDR_DE
MODE IS CHAIN

SET NAME IS ALL_UTIL
MODE IS CALC
MEMBER IS UTILISATEUR
KEY IS JR_UTIL

SET NAME IS ALL_REL
MODE IS CALC
MEMBER IS REL_BAR_CODE
KEY IS JRBAR_VALCODE

SET NAME IS LIEN_RDC
MODE IS CHAIN

SET NAME IS LIEN_REL_LIB
MODE IS CHAIN

SET NAME IS ALL_LIB_BAR
MODE IS CALC
MEMBER IS LIB_BAR
KEY IS VAL_LIB_BAR

SET NAME IS ALL_LIEN_BAR
MODE IS CALC
MEMBER IS LIEN_BAR
KEY IS JR_BAR_LIB

SET NAME IS ALL_LIB_REL
MODE IS CALC
MEMBER IS LIEN_REL
KEY IS REL_COMP_LBR

ANNEXE 3 :

Les écrans.

VOICI UNE EXECUTION DE LA GESTION INTERACTIVE DE BAREMES. ON A ESSAYE
DE LA FAIRE AUSSI COMPLETE QUE POSSIBLE AFIN QUE LE LECTEUR SE
RENDE COMPTE LE MEUX POSSIBLE DES DIFFERENTES POSSIBILITES EXISTANTES.

```
*****
*                                     *
*  SYSTEME DE GESTION DE BAREMES  *
*                                     *
*****
```

Memoire pour le Credit Communal de Bruxelles.

Systeme Cree Par E.EVRARD, le 10/04/1986

Non d'utilisateur : EVRARD

Mot de passe :

BONJOUR, EVRARD
bienvenue sur GESTBAR !
Bon travail!
Taper RETURN :

* MENU DU PROGRAMME *

- 0/ Sortir Du Programme.
- 1/ Creer Un Bareme.
- 2/ Creer Des Valeurs De Bareme.
- 3/ Obtenir Le(s) Libelle(s) D'une Valeur De Code.
- 4/ Afficher Un Bareme.
- 5/ Obtenir La Composition D'un Bareme.
- 6/ Obtenir Les Noms De Tous Les Baremes.
- 7/ Recherche selon critere.
- 8/ Obtenir les acces autorises.
- 9/ Nettoyage du systeme.

Votre reponse ? 1

OPTION 1: Creation le bareme.

Identificateur du bareme : IS0007
Cet identifiant existe deja !
Identificateur du bareme : IS0009

OPTION 1: Creation de bareme.

Date de creation JJ/MM/AA : 10/03/1986
 Type des valeurs le code : AB
 Longueur maximum (00<val<07): 06
 Nbre de version (-1<val<21): 01
 Nombre le(s) libelle(s) (>0) : 02
 Libelle :
 Vous devez entrer une valeur !!
 Libelle : FR
 Reg.ling : FR
 Ce libelle existe deja. Donnez en un autre! Merci!
 Libelle : LOCALITES
 Reg.ling : FR
 Libelle : BAREME DES LOCALITES
 Reg.ling : F
 Val.poss : AB pour anglais,
 FR pour francais,
 NR pour neerlandais,
 AL pour allemand.
 Reg.ling : FR
 Nbre de bar(s) composant(s) (00<=val<10) : 00
 Duree de retention (0000<val): 0300

Votre bareme est cree !

* MENU DU PROGRAMME *

- 0/ Sortir Du Programme.
- 1/ Creer Un Bareme.
- 2/ Creer Des Valeurs De Bareme.
- 3/ Obtenir Le(s) Libelle(s) D'une Valeur De Code.
- 4/ Afficher Un Bareme.
- 5/ Obtenir La Composition D'un Bareme.
- 6/ Obtenir Les Noms De Tous Les Baremes.
- 7/ Recherche selon critere.
- 8/ Obtenir les acces autorises.
- 9/ Nettoyage du systeme.

Votre reponse ? 1

OPTION 1: Creation le bareme.

Identificateur du bareme : IS1012

OPTION 1: Creation le bareme.

Date de creation JJ/MM/AA : 12/03/1986
Type des valeurs le code : FI
Longueur maximo (00<val<07): 02
Nb de version (-1<val<21): 01
Nombre le(s) libelle(s) (>0) : 01
Libelle : BARRE DES TAUX D'INTERETS A APPLIQUER EN COMPTE
Reg.ling : FR
Nb de par(s) composant(s) (00<=val<10) : 04

OPTION 1: Creation de bareme.

Entrez le(s) bareme(s) composant(s) !
Nom du bareme : TOT1

Bareme inexistant !

OPTION 1: Creation le bareme.

Entrez le(s) bareme(s) composant(s) !
Nom du bareme : TYPES DE COMPTES
Role du composant : LIRE TYPE DE COMPTE

OPTION 1: Creation de bareme.

Entrez le(s) bareme(s) composant(s) !
Nom du bareme : TYPES DE COMPTES

Ce bareme est deja composant !

OPTION 1: Creation le bareme.

Entrez le(s) bareme(s) composant(s) !
Nom du bareme : BAREME DES COMPTES

Violation de contrainte !
Ce bareme est compose !
Or un bareme composant ne peut etre que simple !

OPTION 1: Creation le bareme.

Entrez le(s) bareme(s) composant(s) !
Nom du bareme : TYPES D'INTERETS
Role du composant : LIEN INTERETS

OPTION 1: Creation le bareme.

Entrez le(s) bareme(s) composant(s) !
Nom du bareme : TAUX D'INTERETS
Role du composant : LIEN TAUX

OPTION 1: Creation le bareme.

Entrez le(s) bareme(s) composant(s) !
Nom du bareme : CODES TAUX
Role du composant : LIEN CODE TAUX
Duree de retention (0000<val): 0250

Votre bareme est cree !

* FIN DU PROGRAMME *

- 0/ Sortir Du Programme.
- 1/ Creer Un Bareme.
- 2/ Creer Des Valeurs De Bareme.
- 3/ Obtenir Le(s) Libelle(s) D'une Valeur De Code.
- 4/ Afficher Un Bareme.
- 5/ Obtenir La Composition D'un Bareme.
- 6/ Obtenir Les Noms De Tous Les Baremes.
- 7/ Recherche selon critere.
- 8/ Obtenir les acces autorises. 9/ Nettoyage du systeme.

Votre reponse ? 2

OPTION 2: Creation de valeurs de bareme.

Nom du bareme : LOCALITES

OPTION 2: Creation de valeurs de bareme.

Valeur de code : BINCHES

OPTION 2: Creation de valeurs de bareme.

Nombre de(s) libelle(s) (>0) : 01

OPTION 2: Creation de valeurs de bareme.

Libelle : VILLE DE BINCHES
Reg.ling : FR

OPTION 2: Creation de valeurs de bareme.

Date de creation JJ/MM/AA : 19/08/1985
Date de debut d'utilisation : 12/03/1986
Heure de debut d'utilisation : 16130
Votre heure est anterieure
a l'heure courante!
Heure de debut d'utilisation : 16135
Date de fin d'utilisation : 1979
Valeur non numerique !
Annee invalide !
Et/ou mois invalide !
Date de fin d'utilisation : 20/10/1999

Fin de creation de valeur de bareme !

Une autre creation ? NON

* MENU DU PROGRAMME *

- 0/ Sortir Du Programme.
- 1/ Creer Un Bareme.
- 2/ Creer Des Valeurs De Bareme.
- 3/ Obtenir le(s) Libelle(s) D'une Valeur De Code.
- 4/ Afficher Un Bareme.
- 5/ Obtenir La Composition D'un Bareme.
- 6/ Obtenir Les Mons De Tous Les Baremes.
- 7/ Recherche selon critere.
- 8/ Obtenir les acces autorises. 9/ Nettoyage du systeme.

Votre reponse ? 2

OPTION 2: Creation de valeurs de bareme.

Nom du bareme : BAREME DES TAUX D'INTERETS A APPLIQUER EN COMPTE

OPTION 2: Creation de valeurs de bareme.

Valeur de code : 1

OPTION 2: Creation de valeurs de bareme.

Nombre de(s) libelle(s) (>0) : 01

OPTION 2: Creation de valeurs de bareme.

Libelle : LIGNE UN
Reg.lign : FR

OPTION 2: Creation de valeurs de bareme.

Libelle : TYPE COMPTE
Libelle : TYPES DE COMPTES

Valeur le code composante : CV
Role de cette composition : LIEN TYPE COMPTE

OPTION 2: Creation de valeurs de bareme.

Libelle : TPE INTERET
Libelle : TYPES D'INTERETS

Valeur le code composante : D3
Role de cette composition : LIEN TYPE INTERET

OPTION 2: Creation de valeurs de bareme.

Libelle : TX INTERET
Libelle : TAUX D'INTERETS

Valeur le code composante : 6 %
Role de cette composition : LIEN TAUX

OPTION 2: Creation de valeurs de bareme.

Libelle : CODES TAUX

Valeur de code composante : 0
Role de cette composition : LIEU CODE TAUX
Date de creation JJ/MM/AA : 19/08/1986
Date de debut d'utilisation : 19/08/1986
Heure de debut d'utilisation : 16H36
Date de fin d'utilisation : 19/08/1987

Fin de creation de valeur de bareme !

Une autre creation ? N

* MENU DU PROGRAMME *

- 0/ Sortir Du Programme.
- 1/ Creer Un Bareme.
- 2/ Creer Des Valeurs De Bareme.
- 3/ Obtenir Le(s) Libelle(s) D'une Valeur De Code.
- 4/ Afficher Un Bareme.
- 5/ Obtenir La Composition D'un Bareme.
- 6/ Obtenir Les Noms De Tous Les Baremes.
- 7/ Recherche selon critere.
- 8/ Obtenir les acces autorises.
- 9/ Nettoyage du systeme.

Votre reponse ? 3

OPTION 3: Obtention de(s) libelle(s) d'une valeur de code.

Nom du bareme : H043

Bareme inexistant !

* MENU DU PROGRAMME *

- 0/ Sortir Du Programme.
- 1/ Créer Un Barème.
- 2/ Créer Des Valeurs De Barème.
- 3/ Obtenir Le(s) Libelle(s) D'une Valeur De Code.
- 4/ Afficher Un Barème.
- 5/ Obtenir La Composition D'un Barème.
- 6/ Obtenir Les Noms De Tous Les Barèmes.
- 7/ Recherche Selon critère.
- 8/ Obtenir les accès autorisés.
- 9/ Nettoyage du système.

Votre réponse ? 3

OPTION 3: Obtention de(s) libelle(s) d'une valeur de code.

Nom du barème : CODES TAUX

OPTION 3: Obtention de(s) libelle(s) d'une valeur de code.

Valeur de code : 0

OPTION 3: Obtention de(s) libelle(s) d'une valeur de code.

Cette valeur de code est simple !

Valeur de code : 0

Libelle : CODE TAUX NORMAL

Date de creation JJ/MM/AA : 12/03/1986
Date de debut d'utilisation : 12/03/1986
Heure de debut d'utilisation : 15H20
Date de fin d'utilisation : 12/03/1987

* MENU DU PROGRAMME *

- 0/ Sortir Du Programme.
- 1/ Creer Un Barème.
- 2/ Creer Des Valeurs De Barème.
- 3/ Obtenir Le(s) Libelle(s) D'une Valeur De Code.
- 4/ Afficher Un Barème.
- 5/ Obtenir La Composition D'un Barème.
- 6/ Obtenir Les Noms De Tous Les Barèmes.
- 7/ Recherche selon critere.
- 8/ Obtenir les acces autorises.
- 9/ Nettoyage du systeme.

Votre reponse ? 3

OPTION 3: Obtention de(s) libelle(s) d'une valeur de code.

Nom du barème : CODES TAUX

OPTION 3: Obtention de(s) libelle(s) d'une valeur de code.

Valeur de code : 1

OPTION 3: Obtention de(s) libelle(s) d'une valeur de code.

Cette valeur de code est simple !

Valeur le code : 1

Libelle : CODES TAUX SANS INTERET

Date de creation JJ/MM/AA : 12/03/1986
Date de debut d'utilisation : 12/03/1986
Heure de debut d'utilisation : 15H20
Date de fin d'utilisation : 12/03/1987

* MENU DU PROGRAMME *

- 0/ Sortir Du Programme.
- 1/ Creer Un Bareme.
- 2/ Creer Des Valeurs De Bareme.
- 3/ Obtenir Le(s) Libelle(s) D'une Valeur De Code.
- 4/ Afficher Un Bareme.
- 5/ Obtenir La Composition D'un Bareme.
- 6/ Obtenir Les Noms De Tous Les Baremes.
- 7/ Recherche selon critere.
- 8/ Obtenir les acces autorises.
- 9/ Nettoyage du systeme.

Votre reponse ? 3

OPTION 3: Obtention de(s) libelle(s) d'une valeur de code.

Nom du Bareme : BAREME DES COMPTES

OPTION 3: Obtention de(s) libelle(s) d'une valeur de code.

Valeur de code : 250

OPTION 3: Obtention de(s) libelle(s) d'une valeur de code.

Cette valeur de code est composee !

Valeur de code : 250

Libelle : COMPTE NR 063/0000250/10

Date de creation JJ/MM/AA : 19/03/1986
Date de debut d'utilisation : 19/03/1986
Heure de debut d'utilisation : 16H02
Date de fin d'utilisation : 19/03/1987

Liste des valeurs composantes :

Libelle : NOM
Libelle : NOMS DE PERSONNES

Cette valeur de code est simple !

Valeur de code : EVRARD

Libelle : MONSIEUR FREDDY EVRARD

Date de creation JJ/MM/AA : 19/08/1986
Date de debut d'utilisation : 19/08/1986
Heure de debut d'utilisation : 15H10
Date de fin d'utilisation : 19/08/1999

TAPER RETURN !

Liste des valeurs composantes :

Libelle : TYPE COMPTE
Libelle : TYPES DE COMPTES

Cette valeur de code est simple !

Valeur de code : CV

Libelle : COMPTE A VUE
Libelle : ZIEKTREKENING

Date de creation JJ/MM/AA : 19/08/1986
Date de debut d'utilisation : 19/08/1986
Heure de debut d'utilisation : 15H00
Date de fin d'utilisation : 19/08/1987

TAPER RETURN !

Liste des valeurs composantes :

Libelle : TX TOUTES
Libelle : "TAUX D'INTERETS

Cette valeur de code est simple !

Valeur de code : 5 3

Libelle : SIX POUR-CENTS

Date de creation JJ/MM/AA : 19/03/1986
Date de debut d'utilisation : 19/03/1986
Heure de debut d'utilisation : 15H10
Date de fin d'utilisation : 19/03/1987

TAPER RETURN !

Liste des valeurs composantes :

Libelle : CODES TAUX

Cette valeur de code est simple !

Valeur de code : 0

Libelle : CODE TAUX NORMAL

Date de creation JJ/MM/AA : 19/03/1986
Date de debut d'utilisation : 19/03/1986
Heure de debut d'utilisation : 15H20
Date de fin d'utilisation : 19/03/1987

TAPER RETURN !

Liste des valeurs composantes :

Libelle : SLD CREDIT
Libelle : SOLDES "CREDITEUR"

Cette valeur de code est simple !

Valeur de code : 0

Libelle : SOLDES JUL

Date de creation JJ/MM/AA : 19/08/1986
Date de debut d'utilisation : 19/03/1986
Heure de debut d'utilisation : 15H30
Date de fin d'utilisation : 19/08/1987

TAPER RETURN !

Liste des valeurs composantes :

Libelle : SLD DEBIT
Libelle : SOLDES DEBITEURS

Cette valeur de code est simple !

Valeur de code : 30 HFR

Libelle : TRENTE MILLE FRANCS

Date de creation JJ/MM/AA : 19/08/1986
Date de debut d'utilisation : 19/03/1986
Heure de debut d'utilisation : 15H40
Date de fin d'utilisation : 19/08/1987

TAPER RETURN !

* MENU DU PROGRAMME *

- 0/ Sortir Du Programme.
- 1/ Creer Un Bareme.
- 2/ Creer Des Valeurs De Bareme.
- 3/ Obtenir Le(s) Libelle(s) D'une Valeur De Code.
- 4/ Afficher Un Bareme.
- 5/ Obtenir La Composition D'un Bareme.
- 6/ Obtenir Les Noms De Tous Les Baremes.
- 7/ Recherche selon critere.
- 8/ Obtenir les acces autorises.
- 9/ Nettoyage du systeme.

Votre reponse ? 4

OPTION 4: Afficher un bareme.

Nom du bareme : NOM

OPTION 4: Afficher un bareme.

CE BAREME EST SIMPLE !

Son createur est EVRARD .

Libelle : NOM

Libelle : NOMS DE PERSONNES

Identificateur du bareme : IS0007

Date de creation JJ/MM/AA : 19/08/1986

Type des valeurs le code : Alphabetique

Longueur maximum (00<val<07): 06

Nbre de version (-1<val<21): 01

Duree de retention (0000<val): 0300

TAPER RETURN !

NOI

CODE

DURANT

LEMERCE

AIDRE

DUPOND

BEREIN

TROCHU

FRANC

ALBERT

TAPER RETURN !

NOI

CODE

EVBARO

MICHEL

TAPER RETURN !

* MENU DU PROGRAMME *

- 0/ Sortir Du Programme.
- 1/ Creer Un Bareme.
- 2/ Creer Des Valeurs De Bareme.
- 3/ Obtenir Le(s) Libelle(s) D'une Valeur De Code.
- 4/ Afficher Un Bareme.
- 5/ Obtenir La Composition D'un Bareme.
- 6/ Obtenir Les Noms De Tous Les Baremes.
- 7/ Recherche selon critere.
- 8/ Obtenir les acces autorises.
- 9/ Nettoyage du systeme.

Votre reponse ? 4

OPTION 4: Afficher un bareme.

Nom du bareme : BAREME DES COMPTES

OPTION 4: Afficher un bareme.

CE BAREME EST COMPOSE :

Son createur est EVRARD .

Libelle : BAREME DES COMPTES

Identificateur du bareme : IS0003
Date de creation JJ/MM/AA : 19/03/1986
Type des valeurs de code : Numerique
Longueur maximum (00<val<07): 05
Nbre de version (-1<val<21): 01
Duree de retention (0000<val): 0245

TAPER RETURN !

Voulez - vous voir les entetes
des baremes composants <O ou H> ? OUI

Liste des baremes composants :

CE BAREME EST SIMPLE !

Son createur est EVRARD .

Libelle : "O"
Libelle : "OVS DE PERSONNES

Identificateur du bareme : IS1007
Date de creation JJ/MM/AA : 12/03/1986
Type des valeurs de code : Alphanumerique
Longueur maximum (00<val<07): 06
Nbre de version (-1<val<21): 01
Duree de retention (0000<val): 0300

TAPER RETURN !

Liste des baremes composants :

CE BAREME EST SIMPLE !

Son createur est EVRARD .

Libelle : TYPE COMPTES
Libelle : TYPES DE COMPTES

Identificateur du bareme : IS0001
Date de creation JJ/MM/AA : 19/08/1986
Type des valeurs de code : Alphanumerique
Longueur maximum (00<val<07): 02
Nbre de version (-1<val<21): 01
Duree de retention (0000<val): 0300

TAPER RETURN !

Liste des baremes composants :

CE BAREME EST SIMPLE !

Son createur est EVRARD .

Libelle : TX INTERET

Libelle : TAUX D'INTERETS

Identificateur du bareme : IS0003

Date de creation JJ/MM/AA : 19/03/1986

Type des valeurs de code : Alphanumerique

Longueur maximum (00<val<07): 06

Nbre de version (-1<val<21): 01

Duree de retention (0000<val): 0250

TAPER RETURN !

Liste des baremes composants :

CE BAREME EST SIMPLE !

Son createur est EVRARD .

Libelle : CODES TAUX

Identificateur du bareme : IS0004

Date de creation JJ/MM/AA : 12/08/1986

Type des valeurs de code : Numerique

Longueur maximum (00<val<07): 01

Nbre de version (-1<val<21): 01

Duree de retention (0000<val): 0300

TAPER RETURN !

Liste des barenes composants :

CE BARENE EST SIMPLE !

Son createur est EVRARD .

Libelle : SLD CREDIT
Libelle : SOLDES "CREDITEUR"

Identificateur du barene : IS0005
Date de creation JJ/MM/AA : 10/08/1986
Type les valeurs le code : Alphanumerique
Longueur maximum (00<val<07): 06
Nbre de version (-1<val<21): 01
Duree de retention (0000<val): 0245

TAPER RETURN !

Liste des barenes composants :

CE BARENE EST SIMPLE !

Son createur est EVRARD .

Libelle : SLD DEBIT
Libelle : SOLDES DEBITEURS

Identificateur du barene : IS0006
Date de creation JJ/MM/AA : 10/08/1986
Type les valeurs le code : Alphanumerique
Longueur maximum (00<val<07): 06
Nbre de version (-1<val<21): 01
Duree de retention (0000<val): 0300

TAPER RETURN !

PARTIE DES COMPTES					
CODE	NOM	TYPE COMPTE	TX INTERET	CODES TAUX	SLD CREDIT
251	ANDRE	CV	6 %	0	0
252	EVARD	CP	9 %	2	0
253	TRUCHU	CD	6 %	0	0
250	EVARD	CV	6 %	0	0

Taper D pour voir le tableau de droite ou RETURN pour voir la suite : D

PARTIE DES COMPTES					
SLD DEBIT					
30MIFR					
15MIFR					
30MIFR					
30MIFR					

Taper G pour voir le tableau de gauche ou RETURN pour voir la suite : G

PARTIE DES COMPTES					
CODE	NOM	TYPE COMPTE	TX INTERET	CODES TAUX	SLD CREDIT
251	ANDRE	CV	6 %	0	0
252	EVARD	CP	9 %	2	0
253	TRUCHU	CD	6 %	0	0
250	EVARD	CV	6 %	0	0

Taper D pour voir le tableau de droite ou RETURN pour voir la suite :

* MENU DU PROGRAMME *

- 0/ Sortir Du Programme.
- 1/ Creer Un Bareme.
- 2/ Creer Des Valeurs De Bareme.
- 3/ Obtenir Le(s) Libelle(s) D'une Valeur De Code.
- 4/ Afficher Un Bareme.
- 5/ Obtenir La Composition D'un Bareme.
- 6/ Obtenir Les Noms De Tous Les Baremes.
- 7/ Recherche selon critere.
- 8/ Obtenir les acces autorises.
- 9/ Nettoyage du systeme.

Votre reponse ? 5

OPTION 5: Obtention de la composition d'un bareme.

Nom du bareme : BAREME DES COMPTES

OPTION 5: Obtention de la composition d'un bareme.

CE BAREME EST COMPOSE :

Libelle : BAREME DES COMPTES

Liste des baremes composants :

- Libelle : NOM
- Libelle : NOMS DE PERSONNES
- Libelle : TYPE COMPT
- Libelle : TYPES DE COMPTES
- Libelle : TX INTERET
- Libelle : TAUX D'INTERETS
- Libelle : CODES TAUX
- Libelle : SLD CREDIT
- Libelle : SOLDES "CREDITEUR"
- Libelle : SLD DEBIT
- Libelle : SOLDES DEBITEURS

* MENU DU PROGRAMME *

- 0/ Sortir Du Programme.
- 1/ Creer Un Bareme.
- 2/ Creer Des Valeurs De Bareme.
- 3/ Obtenir Le(s) Libelle(s) D'une Valeur De Code.
- 4/ Afficher Un Bareme.
- 5/ Obtenir La Composition D'un Bareme.
- 6/ Obtenir Les Noms De Tous Les Baremes.
- 7/ Recherche selon critere.
- 8/ Obtenir les acces autorises.
- 9/ Nettoyage du systeme.

Votre reponse ? 5

OPTION 5: Obtention de la composition d'un bareme.

Nom du bareme : TYPES DE COMPTES

OPTION 5: Obtention de la composition d'un bareme.

CE BAREME EST SIMPLE !

Libelle : TYPE COMPTE
Libelle : TYPES DE COMPTES

Liste des baremes qui l'admettent comme composant :

Libelle : BAREME DES TAUX D'INTERETS A APPLIQUER EN COMPTE
Libelle : BAREME DES COMPTES

* MENU DU PROGRAMME *

- 0/ Sortir Du Programme.
- 1/ Creer Un Bareme.
- 2/ Creer Des Valeurs De Bareme.
- 3/ Obtenir Le(s) Libelle(s) D'une Valeur De Code.
- 4/ Afficher Un Bareme.
- 5/ Obtenir La Composition D'un Bareme.
- 6/ Obtenir Les Noms De Tous Les Baremes.
- 7/ Recherche selon critere.
- 8/ Obtenir les acces autorises.
- 9/ Nettoyage du systeme.

Votre reponse ? 5

OPTION 5: Obtention de la composition d'un bareme.

Nom du bareme : LOCALITES

OPTION 5: Obtention de la composition d'un bareme.

CE BAREME EST SIMPLE !

Libelle : LOCALITES
Libelle : BAREME DES LOCALITES

Ce bareme n'est pas encore utilise comme composant !

* MENU DU PROGRAMME *

- 0/ Sortir Du Programme.
- 1/ Creer Un Bareme.
- 2/ Creer Des Valeurs De Bareme.
- 3/ Obtenir Le(s) Libelle(s) D'une Valeur De Code.
- 4/ Afficher Un Bareme.
- 5/ Obtenir La Composition D'un Bareme.
- 6/ Obtenir Les Noms De Tous Les Baremes.
- 7/ Recherche selon critere.
- 8/ Obtenir les acces autorises.
- 9/ Nettoyage du systeme.

Votre reponse ? 6

OPTION 6: Voici le nom des baremes :

- Libelle : TX INTERET
 - Libelle : TAUX D'INTERETS
 - Libelle : SLD DEBIT
 - Libelle : SOLDES DEBITEURS
 - Libelle : SLD CREDIT
 - Libelle : SOLEDS "CREDITEUR"
 - Libelle : BAREME DES TAUX D'INTERETS A APPLIQUER EN COMPTE
- Tapez 3 pour stopper !

OPTION 6: Voici le nom des baremes :

- Libelle : BAREME DES COMPTES
 - Libelle : TYPE COMPTE
 - Libelle : TYPES DE COMPTES
 - Libelle : NOM
 - Libelle : NOMS DE PERSONNES
 - Libelle : CODES TAUX
- Tapez 3 pour stopper !

OPTION 6: Voici le non les baremes :

Libelle : TPE INTERET
Libelle : TYPES D'INTERETS

Libelle : LOCALITES
Libelle : BARRE DES LOCALITES

Tapez 5 pour stopper !

* MENU DU PROGRAMME *

- 0/ Sortir Du Programme.
- 1/ Creer Un Bareme.
- 2/ Creer Des Valeurs De Bareme.
- 3/ Obtenir Le(s) Libelle(s) D'une Valeur De Code.
- 4/ Afficher Un Bareme.
- 5/ Obtenir La Composition D'un Bareme.
- 6/ Obtenir Les Noms De Tous Les Baremes.
- 7/ Recherche selon critere.
- 8/ Obtenir les acces autorises.
- 9/ Nettoyage du systeme.

Votre reponse ? 3

OPTION 8: ACCES AUTORISES.

Libelle : BARRE DES TAUX D'INTERETS A APPLIQUER EN COMPTE
Libelle : LOCALITES
Libelle : BARRE DES LOCALITES

Libelle : BARRE DES COMPTES

Libelle : NOM
Libelle : NOMS DE PERSONNES

Tapez 5 pour stopper !

OPTION 8: ACCES AUTORISES.

Libelle : SLD DEBIT
Libelle : SOLDES DEBITEURS

Libelle : SLD CREDIT
Libelle : SOLDES "CREDITEUR"

Libelle : CODES TAUX

Libelle : TX INTERET
Libelle : TAUX D'INTERETS

Tapez S pour stopper !

OPTION 8: ACCES AUTORISES.

Libelle : TPE INTERET
Libelle : TYPES D'INTERETS

Libelle : TYPE COMPTE
Libelle : TYPES DE COMPTES

Tapez S pour stopper !

* MENU DU PROGRAMME *

- 0/ Sortir Du Programme.
- 1/ Creer Un Bareme.
- 2/ Creer Des Valeurs De Bareme.
- 3/ Obtenir Le(s) Libelle(s) D'une Valeur De Code.
- 4/ Afficher Un Bareme.
- 5/ Obtenir La Composition D'un Bareme.
- 6/ Obtenir Les Noms De Tous Les Baremes.
- 7/ Recherche selon critere.
- 8/ Obtenir les acces autorises.
- 9/ Nettoyage du systeme.

Votre reponse ?

OPTION 9: Nettoyage du systeme .

Je passe en revue le bareme compose Nr ISO012 !
Je passe en revue le bareme compose Nr ISO008 !
Je passe en revue le bareme simple Nr ISO003 !
Je passe en revue le bareme simple Nr ISO006 !
Je passe en revue le bareme simple Nr ISO005 !
Je passe en revue le bareme simple Nr ISO001 !
Je passe en revue le bareme simple Nr ISO007 !
Je passe en revue le bareme simple Nr ISO004 !
Je passe en revue le bareme simple Nr ISO002 !
Je passe en revue le bareme simple Nr ISO009 !
Aucun bareme n'est a supprimer!

* MENU DU PROGRAMME *

- 0/ Sortir Du Programme.
- 1/ Creer Un Bareme.
- 2/ Creer Des Valeurs De Bareme.
- 3/ Obtenir Le(s) Libelle(s) D'une Valeur De Code.
- 4/ Afficher Un Bareme.
- 5/ Obtenir La Composition D'un Bareme.
- 6/ Obtenir Les Noms De Tous Les Baremes.
- 7/ Recherche selon critere.
- 8/ Obtenir les acces autorises. 9/ Nettoyage du systeme.

Votre reponse ? 7

OPTION 7: RECHERCHE SELON CRITERE.

Nom du bareme : TYPE DE COMPTES

Bareme inexistant !

* MENU DU PROGRAMME *

- 0/ Sortir Du Programme.
- 1/ Creer Un Bareme.
- 2/ Creer Des Valeurs De Bareme.
- 3/ Obtenir Le(s) Libelle(s) D'une Valeur De Code.
- 4/ Afficher Un Bareme.
- 5/ Obtenir La Composition D'un Bareme.
- 6/ Obtenir Les Noms De Tous Les Baremes.
- 7/ Recherche selon critere.
- 8/ Obtenir les acces autorises.
- 9/ Nettoyage du systeme.

Votre reponse ? 7

OPTION 7: RECHERCHE SELON CRITERE.

Nom du bareme : TYPES DE COMPTES

OPTION 7: RECHERCHE SELON CRITERE.

Recherche sur critere interdite pour les baremes simples!

* MENU DU PROGRAMME *

- 0/ Sortir Du Programme.
- 1/ Creer Un Bareme.
- 2/ Creer Des Valeurs De Bareme.
- 3/ Obtenir Le(s) Libelle(s) D'une Valeur De Code.
- 4/ Afficher Un Bareme.
- 5/ Obtenir La Composition D'un Bareme.
- 6/ Obtenir Les Mots De Tous Les Baremes.
- 7/ Recherche selon critere.
- 8/ Obtenir les acces autorises.
- 9/ Nettoyage du systeme.

Votre reponse ? 7

OPTION 7: RECHERCHE SELON CRITERE.

Nom du bareme : BAREME DES COMPTES

OPTION 7: RECHERCHE SELON CRITERE.

Libelle : NOM
Libelle : NOMS DE PERSONNES

Ce bareme fait-il partie de la recherche ? 0

OPTION 7: RECHERCHE SELON CRITERE.

Libelle : TYPE COMPTES
Libelle : TYPES DE COMPTES

Ce bareme fait-il partie de la recherche ? 1

OPTION 7: RECHERCHE SELON CRITERE.

Libelle : TX INTERET
Libelle : TAUX D'INTERETS

Ce bareme fait-il partie de la recherche ? N

OPTION 7: RECHERCHE SELON CRITERE.

Libelle : CODES TAUX

Ce bareme fait-il partie de la recherche ? N

OPTION 7: RECHERCHE SELON CRITERE.

Libelle : SLD CREDIT
Libelle : SOLDES "CREDITEUR"

Ce bareme fait-il partie de la recherche ? N

OPTION 7: RECHERCHE SELON CRITERE.

Libelle : SLD DEBIT
Libelle : SOLDES DEBITEURS

Ce bareme fait-il partie de la recherche ? N

OPTION 7: RECHERCHE SELON CRITERE.

Libelle : NOM
Libelle : NOMS DE PERSONNES

Valeur de code composante : EVRARD

BAREME DES COMPTES

CODE	NOM	TYPE COMPTE	TX INTERET	CODES TAUX	SLD CREDIT
252	EVARD	CP	9 %	2	0
250	EVARD	CV	6 %	0	0

Taper D pour voir le tableau de droite ou RETURN pour voir la suite :D

BAREME DES COMPTES

SLD DEBIT

15MIFR

30MIFR

Taper G pour voir le tableau de gauche ou RETURN pour voir la suite :G

BAREME DES COMPTES

CODE	NOM	TYPE COMPTE	TX INTERET	CODES TAUX	SLD CREDIT
252	EVARD	CP	9 %	2	0
250	EVARD	CV	6 %	0	0

Taper D pour voir le tableau de droite ou RETURN pour voir la suite :

* MENU DU PROGRAMME *

- 0/ Sortir Du Programme.
- 1/ Creer Un Bareme.
- 2/ Creer Des Valeurs De Bareme.
- 3/ Obtenir Le(s) Libelle(s) D'une Valeur De Code.
- 4/ Afficher Un Bareme.
- 5/ Obtenir La Composition D'un Bareme.
- 6/ Obtenir Les Noms De Tous Les Baremes.
- 7/ Recherche selon critere.
- 8/ Obtenir les acces autorises.
- 9/ Nettoyage du systeme.

Votre reponse ? 7

OPTION 7: RECHERCHE SELON CRITERE.

Nom du bareme : BAREME DES COMPTES

OPTION 7: RECHERCHE SELON CRITERE.

Libelle : NOM
Libelle : NOMS DE PERSONNES

Ce bareme fait-il partie de la recherche ? 0

OPTION 7: RECHERCHE SELON CRITERE.

Libelle : TYPE COMPTE
Libelle : TYPES DE COMPTES

Ce bareme fait-il partie de la recherche ? 1

OPTION 7: RECHERCHE SELON CRITERE.

Libelle : TX INTERET
Libelle : TAUX D'INTERETS

Ce bareme fait-il partie de la recherche ? N

OPTION 7: RECHERCHE SELON CRITERE.

Libelle : CODES TAUX

Ce bareme fait-il partie de la recherche ? N

OPTION 7: RECHERCHE SELON CRITERE.

Libelle : SLD CREDIT
Libelle : SOLDES "CREDITEUR"

Ce bareme fait-il partie de la recherche ? N

OPTION 7: RECHERCHE SELON CRITERE.

Libelle : SLD DEBIT
Libelle : SOLDES DEBITEURS

Ce bareme fait-il partie de la recherche ? N

OPTION 7: RECHERCHE SELON CRITERE.

Libelle : NOM
Libelle : NOMS DE PERSONNES

Valeur de code composante : MICHEL

Aucune ligne de ce bareme ne correspond a votre critere!

* MENU DU PROGRAMME *

- 0/ Sortir Du Programme.
- 1/ Creer Un Bareme.
- 2/ Creer Des Valeurs De Bareme.
- 3/ Obtenir Le(s) Libelle(s) D'une Valeur De Code.
- 4/ Afficher Un Bareme.
- 5/ Obtenir La Composition D'un Bareme.
- 6/ Obtenir Les Noms De Tous Les Baremes.
- 7/ Recherche selon critere.
- 8/ Obtenir les acces autorises.
- 9/ Nettoyage du systeme.

Votre reponse ? 0
